(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 August 2000 (24.08.2000)

PCT

(10) International Publication Number
WO 00/49571 A3

(51) International Patent Classification[7]: G06T 9/00, 9/40

(21) International Application Number: PCT/CA00/00134

(22) International Filing Date: 15 February 2000 (15.02.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2,261,833 15 February 1999 (15.02.1999) CA

(71) Applicant *(for all designated States except US)*: **DIGITAL ACCELERATOR CORPORATION** [CA/CA]; Suite 840, 650 West Georgia Street, Vancouver, British Columbia V6B 4N9 (CA).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **WANG, Meng** [CA/CA]; 3027 Laurel Street, Vancouver, British Columbia V5Z 3T6 (CA). **YANG, Xue, Dong** [CA/CA]; 3446 Rideout Bay, Regina, Saskatchewan S4S 7C3 (CA). **SIMON, Brent** [CA/CA]; 210 West Keith Road, North Vancouver, British Columbia V7P 1Z3 (CA). **QU, Li** [CA/CA]; 318-165 East 6th Street, North Vancouver, British Columbia V7L 1P2 (CA). **XIONG, Yi** [CA/CA]; 328 Louis Riel House, SFU, Vancouver, British Columbia V5A 1S6 (CA). **WONG, Michael** [CA/CA]; 7831 Welsley Drive, Burnaby, British Columbia V5E 3X4 (CA).

(74) Agent: **MBM & CO.**; P.O. Box 809, Station B, Ottawa, Ontario K1P 5P9 (CA).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

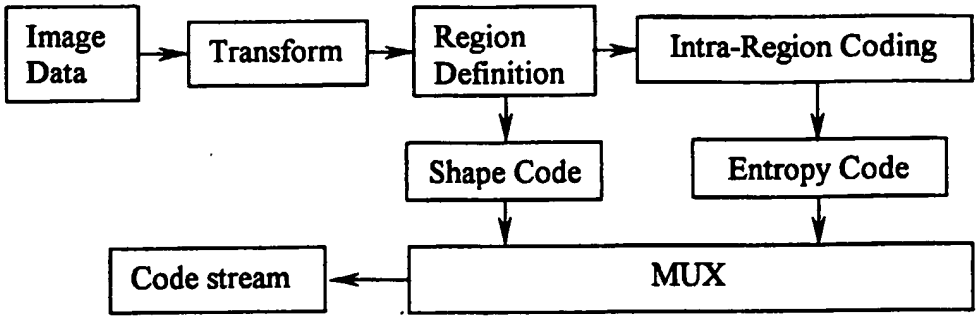
(88) Date of publication of the international search report:
5 April 2001

(48) Date of publication of this corrected version:
21 June 2001

(15) Information about Correction:
see PCT Gazette No. 25/2001 of 21 June 2001, Section II

[Continued on next page]

(54) Title: METHOD AND SYSTEM OF REGION-BASED IMAGE CODING WITH DYNAMIC STREAMING OF CODE BLOCKS



(57) Abstract: The invention is a new effective and fast method and apparatus for still image compression, transmission and decompression. The present invention implements optional sorting and encoding methodologies to create object-oriented, shape coding or region coding in still image and video coding that is content accessible, scalable and computationally efficient. The invented, multiplexed transmission regime ensures a robust, scalable and content accessible bit stream and includes a dynamic, bit budget control architecture.

WO 00/49571 A3

METHOD AND SYSTEM OF REGION-BASED IMAGE CODING WITH DYNAMIC STREAMING OF CODE BLOCKS

Field of the Invention

The present invention relates generally to image coding, and more particularly to the compression and streaming of scalable and content-based, randomly accessible digital still images.

Background of the Invention

With the rapid growth of the internet and multimedia applications, there is a great demand for new image coding tools that that will provide for high quality processing capability, an efficient internal architecture, and flexibility in terms of future technological advances. This is a challenge that has been put forth before the JPEG 2000 Committee. Although the main focus should be to provide state-of-the-art compression performance, JPEG 2000 should also offer unprecedented content based accessibility of the compressed format to support applications of various needs. It is highly preferred and advantageous that image content be accessed, manipulated, exchanged, and stored in compact form.

In order for JPEG 2000 to be the standard coding foundation of new generation image processing systems, it must provide for efficiency in coding, different types of still images (bi-level, gray-level, color) with different characteristics (natural images, scientific, medical imagery, rendered graphics, text, etc.) within a unified system. In addition to providing low bit rate operation with quality performance superiority to existing standards, this new system should include many modern features as listed in the JPEG 2000 requirement document.

The open architecture and the set of algorithms presented in this document are based on Digital Accelerator Corporation's (DAC) Region based Image Coding System (RICS). RICS has not only achieved a rate distortion performance competitive with best known compression techniques, but also demonstrates a high degree of openness and flexibility that will accommodate most well known algorithms as well as new yet

to be implemented. [redacted] features supported by RICS covers alm[redacted]ll those listed in JPEG 2000 Requirements document.

Generality

Generality is a primary concern in the architectural design of RICS. The RICS architecture attempts to be an integrated platform that supports and facilitates a variety of applications that may have different characteristics and requirements. For example, providing efficient lossless and lossy compression in a single code stream; efficient processing of compound documents containing both bi-level (text) and color imagery; progressive transmission by pixel accuracy and/or by resolution; random access of arbitrary shaped regions; and so on.

Openness

We also understand that the new JPEG 2000 standard is intended to be a dynamic, rather than static, suite of coding tools that support the new generation imagery applications and, at the same time, keeping abreast with the progress of technology. The mathematical foundation for those leading candidates of new image coding methods (most noticeably the various types of multi-resolution analysis techniques, such as wavelet transforms) is relatively young and still under intensive investigation. If an architectural design is based on or restricted to one or several particular existing coding methods, it may become outdated very quickly.

In attempting to produce a flexible and open platform, the RICS architecture organizes the image coding process into a set of functionally separable modules, such that each module can be developed and optimized individually. Furthermore, all modules in the system are designed to be functionally orthogonal with each other, in the sense that a new algorithm, without affecting the functionality of other modules, can effectively replace the base algorithm of any specific module.

This open architecture will not only be able to accommodate future new algorithms, but also makes compatibility with other standards an easy and natural extension of many concepts, as will be explained later in this document.

Accessibility

Content based accessibility is becoming an important feature in supporting applications such as multimedia database query, internet server-client interaction, content production, remote diagnostics, and interactive entertainment. The content-based accessibility requires that semantically meaningful visual objects be used as basis for image data representation, explanation, manipulation, and exchange. With images being represented in compressed format, it is desirable to perform retrieval operations directly in the code space without requiring image reconstruction. In fact, any search algorithms that require image reconstruction will be infeasible from a practical viewpoint because of the huge amount of images in most image databases. The previous JPEG and many existing coding techniques focus primarily on the issue of compression ratio, paying minimal attention to the need of content based image retrieval.

RICS, by its name, has a fundamental consideration to various types of regions in images. The regions referred to as ROI (Region of Interest) are usually user-specified primitive geometric shapes, such as rectangles or circles. The regions that define the visual objects are usually of arbitrary shapes. Regions can also be generated as the result of certain mathematical operations or transform properties (e.g. significance of transformation coefficients) used to partition the image into disjoint regions of various (e.g. tiles, hierarchical blocks, or arbitrary shapes).

In designing the RICS system, DAC considered carefully the distinction between the concept of an object and that of a region. RICS is open to very general definition of region. A region is a 2D spatial identity with pure syntactic contribution to a code stream. In contrast, an object may contain semantic information. Therefore, the region is perceived as a more elementary and reliable description than the object. RICS is region based, not object based. RICS supports a rich set of region types, from primitive geometrical shapes, tiles, hierarchical blocks, to most general arbitrary shapes.

The region based coding strategy effectively supports the content-based accessibility of imagery data. Specifically, this ability enables random access to the code stream as well as providing a processing channel for user defined regions of interest or tile based techniques. Supporting MPEG-4 object based accessibility is one of the main

objectives of the R[redacted] design. Furthermore, region-based codi[redacted]rovides a natural bridge for 'transcodability' with JPEG and JBIG.

Scalability

Many applications require image data that is available at different resolutions or qualities for decoding. For example, in a progressive transmission process, the bit rate control mechanism should allow the image data to be transmitted in certain priority order, and be able to truncate the remaining data flexibly, either upon request from the receiving terminal or upon channel limitation. Scalable image coding involves generating a coded representation (code stream) in a manner that facilitates the reconstruction of the image at multiple resolutions or quality levels by scalable coding.

Ideally, the control of scalability should be centralized in a single module as the last stage on the encoder side right before the code stream is fed to the communication channel. Furthermore, it is desirable that the scalability control module can completely handle the required processing locally, without any further request propagating back to any previous stages in the encoder. In this way the scalability control module avoids the need for multi-pass computation.

The RICS architecture is designed to support three types of scalability: scalability in terms of pixel precision, spatial resolution, and regions.

Compactness

There is no doubt that the new image coding standard should offer a higher compression performance than the former JPEG, especially at the low bit rate end.

Integrating the compactness, scalability, and accessibility into a general purpose, flexible, and open architecture represents a challenge for JPEG 2000. The RICS is designed to provide a solution. The basic idea of region based coding is as follows:

- The input image data, after certain transformation, becomes a set of image primitives. This set of primitives can be wavelet transform coefficients, DCT coefficients, other transform coefficients, or even raw image data.

- The image primitives are grouped into regions. Region definition can come from user defined ROI, from other application modules, or from certain automatic segmentation algorithms running in the primitive space.

- Each region contains one or more independent coding units (ICU). The primitives in an ICU are encoded and decoded independently, without reference to primitives of any other ICU. This procedure is called the intra-region coding. The outcome of an ICU operation is a code block.

- A multiplexer (MUX) is employed to integrate the code blocks into the final code stream.

A Brief Description of the Drawings

Figure 1 is a simplified RICS block diagram.

Figure 2 is a detailed RICS block diagram.

Figure 3 is subband decomposition schemes.

Figure 4 is a Wavelet Filter Library.

Figure 5 is a performance comparison of YUV and standardized color systems.

Figure 6 is geometric shapes supported by RICS.

Figure 7 is a hierarchical partitioning.

Figure 8 is the relationship between regions and objects.

Figure 9 is threshold masks obtained from Level 1 Lena Wavelet Decomposition.

Figure 10 is a Level 1 common mask obtained by thresholding the combined data set.

Figure 11 is individual region masks obtained by separating raw the common mask.

Figure 12 is individual region masks obtained by separating raw common mask.

Figure 13 is a spectral magnitude of zigzag spectrum as a function of position.

Figure 14 is common mask spectrum filter sizes and captured coefficient numbers.

Figure 15 is a coefficient banding concept used to quantize the common mask spectrum.

Figure 16 is Quantization Band Sizes for Common Mask Spectrum of size 128 by 128.

Figure 17 is Spectral Quantization Band Sizes for Other Common Mask Sizes.

Figure 18 is a Mask Overhead for Grayscale Images.

Figure 19 is a low [redacted]Common Mask Approximation for gener[redacted]efficient classification.

Figure 20 is a Mask Quantization following the Spectral Content for Bit Allocation.

Figure 21 is Translated Common Masks for Remaining Resolution Levels.

Figure 22 is the three types of ICU structures.

Figure 23 is a Pavement of a region using type-1 ICUs.

Figure 24 is a Embedded quad tree flowchart.

Figure 25 is VLC Codes for EQW.

Figure 26 is the transcode with JBIG.

Figure 27 is the 'trans-out' and 'trans-in' modes for transcoding with JBIG.

Figure 28 is Wavelet Mallot Decomposition for Lossy Color Transform Data.

Figure 29 is Level Priority Processing Order for Each Channel (Lossy Case).

Figure 30 is Level Priority Processing Order (Lossy).

Figure 31 is Level Priority Color Interleave Processing Order (Lossy).

Figure 32 is Level Priority Color Processing Order (Lossless).

Figure 33 is Level Priority Color Interleave Processing Order (Lossless).

Figure 34 is Typical MUX List Data Structure.

Figure 35 is SNR Progressive Bit Budget Distribution Scheme.

Figure 36 is MUX List Overhead for Y-Channel 8-Level Decomposition.

Figure 37 is MUX List Overhead for Square Images of Various Sizes.

Figure 38 is MUX List Overhead for Square Images of Various Sizes.

Figure 39 is General Region Level Priority Color Processing Order (Lossy).

Figure 40 is Region Level Priority Color Processing Order (Lossy).

Figure 41 is Color Interleave Region Level Priority Processing Order (Lossy).

Figure 42 is Region Level Priority Color Processing Order (Lossless).

Figure 43 is Transparent Region Level Priority Color Processing Order (Lossy).

Figure 44 is Transparent Region Level Priority Color Processing Order (Lossless).

Figure 45 is a graph representation of the transparent region level in color mode for the 4 DCT region channels.

Figure 46 is a graph representation of the region priority level in color mode for the 4 DCT channels.

Figure 47 is a graph representation of the absolute region priority level color mode over the 4 DCT region channels.

Figure 48 is a graph representation of scaled region priority level per mode of the 4 DCT channels.

Figure 49 is a graph representation of the region percentage priority level in color mode over the 4 DCT region channels.

Figure 50 is a graph representation of the mixed processing multiplexer mode of operation over the 4 DCT channels

Figure 51 is a diagram of the basic lossless header structure.

Figure 52 is a diagram of the basic lossy header structure.

Figure 53 is a diagram of the basic header tag structure.

Figure 54 is a block diagram of the basic bit stream syntax for normal modes of operation in the lossy case.

Figure 55 is a block diagram of basic bit stream syntax for region modes of operation.

Figure 56 is a block diagram of the basic bit stream syntax for mixed modes of operation.

Figure 57 is a block diagram of the JPEG transcode/decode method

Figure 58 is a block diagram of the JBIG transcode/decode method

Figure 59 is a block diagram of the modified post filtering procedure

Figure 60 are three gradations of post processing in digital still images..

Figure 61 is a representation Edge areas where the de-ringing filtering is selectively applies.

Figure 62 is a table of Test fImage: Camera (256x256 grayscale) Test Image: hk (256x256 color)

Figure 63 is a table of Test Image hk.raw results

Figure 64 is a table of Test Image camera.raw (grayscale) results

Figure 65 is a table of Test Image hk.raw (color image)results

Figure 66 is a table of Camera raw (part a) results

Figure 67 is a table of Camera.raw (part a) results

Figure 68 is a table of HK raw results

Figure 69 is a table of: Quality versus iterations.

Figure 70 is a table of: Quality versus iterations.As discussed.

A Detailed Description of the Drawings

The System Architecture

The architecture of the RICS system can be described as the scheme of *dynamic streaming of code blocks*. In this design, the image primitives of ICUs generate unit *code blocks*. A code block is a logically independent unit of coding and decoding which does not rely on information contained in any other blocks. Compression is achieved in each block coder, and the coding efficiency of the block coders determines the overall efficiency of a RICS system. The openness of the system is reflected in the different coding algorithms that can be used to produce different code blocks. The system uses a multiplex structure (a MUX) to assemble the code blocks into the code stream and realize the bit rate allocation. In short, the RICS architecture allows for flexibility in the areas of compactness and openness to the block coders and, at the same time, allowing the multiplexer to handle the various schemes of scalability and random accessibility to the code stream.

It should be noted that the *block* used in RICS is a logical unit rather than a geometric concept. A code block may correspond to an 8x8 tile (in the case of JPEG mode coding), a pyramid data structure in zero tree like coding schemes, a rectangular area in block based coding schemes, or an arbitrary shaped area in the raw image buffer. The independence of encoding and decoding is the primary requirement of a code block. A code block is the outcome of an intra-region coding.

Figure 1 illustrates the simplified RICS coding architecture. A more detailed diagram is shown in Figure 2. A detailed description of the function of each module is given in this document. Because the RICS is intended to be an open architecture, DAC does not limit each module to any specific algorithm. Instead, any individual algorithm can be placed into a module as long as it meets the functional requirement of that module. Supporting algorithms are included for certain modules, which reflect preferred operation of the overall system.

Types of Input Image

As shown in Figure 2, the RICS architecture supports the coding of three types of image data: grayscale, color, and bi-level images.

Transformations

In a typical multi-resolution coding scheme, an image is transformed via a multi-resolution decomposition process. In the proposed architecture, transforms such as KL, wavelet, wavelet package, lifting scheme, etc. can be placed in the transformation module. These transforms produce a set of decomposition coefficients {Cij} at different resolution levels and in different spatial orientations.

The RICS architecture also supports DCT or windowed Fourier transform as a transformation technique. This is mainly for the transcodability with JPEG. It should be noted that the Fourier based transforms have been studied for more than a century; its theory is relatively complete and its mathematical and physical properties are well understood. Particularly, its translation, scaling, and rotation properties may be very useful for content based retrieval computations. On the other hand, wavelet transforms are relatively new, and many of its properties require further investigations. As a result, the support of DCT may have an impact beyond the sole backward compatibility consideration.

RICS also allows the NULL transformation (that is, no transformation is applied at all). In this case, an identity transformation is applied to the raw image data as the transformation step. The NULL transformation is useful in several instances. For example, it is usually not beneficial to apply DCT or wavelet transforms to bi-level images (text) for compression purpose. Another example is the residual images in video coding. The information in a residual image is the difference between video blocks and has a high frequency spectral content already. It is highly questionable whether it is beneficial at all to apply another mathematical decomposition (DCT or wavelet) to this type of data for the purpose of compact coding.

Region Definition

The function of this module is to partition the coefficients produced by the transformation module into a number of spatial regions. The RICS supports three region schemes.

1. Automatic partition based on the preordering of the coefficients.
2. Partition based on user defined ROI or object related shapes.
3. Partition based on tiling.

The first partition scheme is also referred to as the region hierarchy formation process. Consider the transformation coefficients as a set. The region hierarchy formation process partitions the set into a number of hierarchically disjoint subsets according to certain definitions of significance. In providing a very general partitioning technique captured in a very general region based control architecture, RICS can perform highly flexible progressive transmission modes of operation that depend on data priorities set up for the code stream.

Scheme 2 deals with user specified ROI, typically primitive geometrical shapes, such as rectangles or circles, as well as object related shapes. Object related shapes could come from a variety of sources, such as user input, motion analysis in video compression, etc.

Scheme 3 is a simple partition and requires minimal for shape coding. This scheme is essential in the JPEG mode. Some wavelet based compression techniques utilize this scheme to explore coding efficiency. This scheme offers very little support for content based accessibility.

Hierarchically disjoint regions can be used in combination with user defined ROI in still image processing and objects in video processing. However providing a fair user partition for detail information is difficult in still image compression. But automatic partitioning or preordering techniques can be performed to control user selections in both arbitrary and block based modes of operation. The research presented in this document introduces a new multi-level control architecture for advanced multi-level image processing. The ability to perform a host of partitioning and preordering techniques in both normal and region based modes of operation is encompassed in

this architecture. [redacted] arbitrary and automatic region formation [redacted] mes are handled in the same manner at a high level.

Region Shape Coding

Three types of region shape are supported in RICS: tiling, primitive geometric shapes, and arbitrary shapes.

1. Tiling requires only a small set of parameters to describe the configuration especially in sequence based processing modes. However the sequences can be organized and presented in many ways in packing them into the final code stream.

2. Primitive geometrical shapes can also be coded efficiently. For example, a rectangle can be defined by four integers (x_{min}, y_{min}) and (x_{max}, y_{max}), or (x_{min}, y_{min}) and *(width, height)* etc.

3. Arbitrary shapes are more difficult and costly to encode. Partitions produced automatically by image analysis algorithms may contain many small regions. Specifically, the auto-region detection routines presented here produces hierarchically organized data partitions. This presents a highly challenging problem for shape coding. RICS provides two practical solutions to this problem. One is a quad tree based hierarchical region description. This mode of operation follows bit level ordering at different resolution levels (see embedded quad tree wavelet (EQW) in Ch. 4.) The other is a DCT coded multi-level region channel definition followed by preordering the partition (given the restraints of the code stream in terms of overhead). Though the coding efficiency of the second solution is currently slightly poorer than the first, it does contain a number of attractive features:

   - It provides a single representation for multi-level bitmaps.
   - From this single representation, region masks can be reproduced at arbitrary resolution levels, which is useful for subband coding.
   - It generates curved shapes, which potentially (e.g. in case of large number of complex curved regions) could outperform quad tree based region descriptions.

- It has several useful properties, such as translation and rotation properties, which the quad tree based description does not offer. In fact, a quad tree representation will change dramatically if an object is slightly translated in the image, making this type of representation not suitable for content based retrieval applications. However, the main advantage of the quad tree representation is its simplicity and most noticeably in block based modes.

The region shape code is included in the code stream when the region definition is determined at the encoder. In the case of decoder specified regions, the region shape coding has a whole new meaning.

Intra-Region Coding

The function of the intra-region coding is to arrange the transformation data in an arbitrary shaped region into a one dimensional code stream. Regardless of the region definition scheme or the region coding technique, this streaming process requires an intermediate state where a control architecture can be designed to tailor the region channels whether dealing with a bitmap mask, an auto-detection routine or any other of numerous classification techniques. At the decoding end, the inverse routine generates the same mask to unpack the values from the one dimensional code stream and place them back into the correct position in each region.

Intra-region coding is completed in block coders. Different block coders can be used to produce the 1D code stream. For JPEG mode, the zigzag scanning/quantization routine is called to pack an 8x8 DCT block. For wavelet based coding, both explicit quantization and implicit quantization schemes can be used. In particular, embedded zero tree and embedded quad tree can be used as implicit quantization schemes. Furthermore most implicit quantization schemes are implicitly decodable. In dealing with bi-level images, a JBIG routine can be called a block coder. This effect can be staged by calling an implicit quantization scheme using one bit plane. Alternatively, efficient JBIG routines can be called block coders in an embedded coding scheme at each of the multiple bit planes.

The Multiplexer

The function of the multiplexer is to assemble the code blocks derived from different subbands and different regions in proper orders into a single code stream. Due to the

richness of region [redacted]ition and block coding schemes, there a[redacted]lenty of ways to pack the code blocks. Different ways to merging the data lead to different transmission priorities. For all transmission ordering modes, the final code stream can be transmitted progressively, and can be truncated at any desired place.

The notion of using a multiplexer has been adopted in some standardized processes such as MPEG. In contrast, the use of a multiplexer in the design of a still image coding system is rare. The region based coding strategy opens the opportunity to systematically explore the syntactic and semantic richness in code stream ordering and transmission medium. With the image segmentation techniques improving in the future, region shapes will become more accurate in tracking objects in the image. In that case, the multiplexer will not only work as a syntactic composer, but also impose semantic meanings to the code stream.

Highlight of Features

The RICS is a true open architecture. It supports not only the algorithms DAC has developed, but also can accommodate most existing well known compression algorithms. Users may include their own functions that are appropriate to their application in a number of modules such as transformation, region definition, region shape coding, and intra-region coding all under the MUX control architecture. It is also implicitly flexible to new technological advances.

- DAC's current implementation offers superior low bit rate performance that is competitive with best existing compression techniques.
- The richness in region definition in the RICS allows great accessibility, thus providing a solid foundation for content based image applications.
- It covers compression for both continuous tone and bi-level images in a single unified architecture.
- It provides lossy and lossless compression in a single, natural, code stream in the course of progressive decoding.
- It provides a variety of progressive transmission modes that allow images to be reconstructed with increasing pixel accuracy or spatial resolution using region priority modes for user specified ROI or system defined significant areas.

- It supports both variable rate and fixed size modes.
- It supports very flexible random access to and processing for regions with arbitrary shape.
- It provides a graceful backward compatibility (or transcodability) with JPEG.
- The generic region definition in RICS is a very suitable interface for object-based coding schemes currently under development by MPEG-4. (In fact, as the arbitrary region shape begins to fit dynamic objects with more accuracy, motion estimation is more definite, and consequently allows for more efficient error compensation using region-based coding.)
- Our general region shape definition provides a solid basis for object based composition and object based information embedding. Multiple objects with arbitrary shape are accepted.
- Our bit stream is robust to bit errors. Each unit structure used by the MUX is independently decodable.
- Incorporating standardized encryption techniques with the RICS system is straightforward.

Transforms

In order to support multiple application needs, provide transcodability, and accommodate future growth, the RICS architecture is designed to support three categories of transformation: the DCT, wavelet transforms, and a special NULL transform.

DCT

The RICS architecture supports the DCT transforms as defined in the baseline JPEG.

Wavelet Transform

The RICS architecture supports various kinds of subband decomposition schemes, including the three schemes in Figure 3.

The Null Transfor

In dealing with the compound documents with mixed contents, it is sometimes required to encode some areas of the image without any transform. In particular, regions with bi-level pixels usually do not need to be transformed for coding purpose. In coding these regions, the transform stage is bypassed; the effect is simulated by a NULL transform stage in the process.

The present version of RICS uses totally 34 types of wavelet kernels as given in the table in Figure 4. In our experiments, the biorthogonal filters (Bior9-15 of the table) seemed to give the best compression. Both low pass filtering and high pass filtering are done using convolution. After filtering, the wavelet coefficients are down sampled by 2. This process is repeated using the low pass part until the desired decomposition level is reached. In inverse wavelet transform, quadtrature mirror filters (QMF) for low and high pass are used. Then the coefficients are up-sampled by 2.

Color Transform

In addition to YUV format, the RICS system uses the Karhunen-Loeve Transform for color transformation. Following the notation of statistics we term this process as the *color standardization*. The KL color transform requires more computation than YUV transform. Figure 5 shows the PSNR comparison of the two color transforms. More test results are also available from DAC.

Region Definition and Shape Coding

The notion of region processing plays a fundamental role in the operation of RICS. The choice of region based coding is motivated by many application needs such as content based retrieval, interactive multimedia application, graphics object and image composition, and coding of dynamic object in video compression.

There are two fundamental issues in a region based coding strategy: defining regions and coding the regions. In the RICS system, region coding is divided into the shape coding and the intra-region coding (the content coding). Section 3, describes the

various schemes f[redacted]gion definition and shape coding. The [redacted]-region coding is discussed below.

From the viewpoint of JPEG 2000, the process of region definition does not have to be standardized. However, schemes for forming regions that the JPEG 2000 will need to support should be defined clearly. The RICS supports three types of region definition.

- Tile based organization
- primitive geometric representations
- arbitrary shapes

Region definition is an optional step: an image can be coded without specifying any region (the non-region mode). In this case, the entire image area is considered as a single region.

Tiling: Definitions and Shape Coding

Tiling is perhaps the simplest region definition scheme. In this scheme, the entire image area is divided into a number of rectangular blocks. In particular, 8 by 8 tiles are used in the JPEG coding mode. Most tiling approaches cannot cover the natural shapes, region trends or partial objects in a picture.

For coding a tiling scheme, a small set of global parameters will be sufficient. The primary parameter is the size of tiling block and the technology is developed around this theme. In case of subband decomposition, tiling block size may vary from one subband to another. However, the RICS architecture has a sound operational foundation. It can be operated in both tile based or arbitrary processing modes of optional arbitrary region formation schemes.

Primitive Geometrical Shapes

Primitive geometrical shapes are ideal for supporting user interaction; i.e. user specified ROI, and for processing compound documents where the text areas can be well covered with one or more rectangular boxes. Geometric shapes currently supported by RICS are listed in the Table in Figure 6.

Arbitrary Regions [redacted] Generic Binary Partition Hierarchy [redacted]

Let C_0 be the set of image primitives. Given an ordering relationship $\succ$, generic binaries partition of C_0 into C_{10} and C_{11} is defined by the following conditions.

$$\begin{aligned} &(1)\ C_{10} \cup C_{11} = C_0, \\ &(2) C_{10} \cap C_{11} = \Phi, \text{ and} \\ &(3) \text{ for any } a \in C_{10} \text{ and } b \in C_{11},\ a \succ b \text{ holds.} \end{aligned} \qquad \text{(Eq.III.1)}$$

Recursively, each of the new generated subsets can be further partitioned into two subsets, resulting in a hierarchical structure to represent the partition (see Figure 7)

A useful ordering relation popular in the compression community is "more significant than": $a \succ b$ whereby a is more significant than b. The definition of significance can be very general. One definition may be used to create an ordering, or several definitions are combined together to generate an ordering. This can be illustrated by the following example shown in Figure 8. Suppose the region masks shown in the leftmost image are generated by considering the magnitude of wavelet transform coefficients as the measure of significance. The region mask in the middle image is an object shape generated by another definition of importance that, for example, can be the intensive dynamic region in a video frame. The rightmost image shows the intersection of the two region masks, which is a new region partitioned according to a multiple definition of significance.

As an example, if the significance is measured by magnitude of wavelet transform coefficients, then the ordering relation is simply the numeric ">" relation. To create a binary partition, a set of thresholds is sufficient. The first threshold, say T_0, separate the initial set C_0 into two subsets, C_{10} and C_{11}. Subsequent threshold values can be specified to recursively partition the new subsets.

The above partition scheme produces accurately a number of disjoint subsets which satisfies the strict ordering relationship of $\succ$. However, it usually produces many small, scattered regions. Consequently, the representation of region shapes for these

subsets can be pot[redacted]lly expensive. From the representation [redacted]t of view, fewer numbers of smoothly shaped regions are desirable. In the following, a less strict partition scheme is defined, which takes the above partition scheme as a special case.

Instead of requiring the condition $a \succ b$ to be held for all $a \in C_{10}$, $b \in C_{11}$, it is replaced by a looser condition

$$\neg b \in C_1,\ b \succ a, \text{ for any } a \in C_0. \quad \text{(Eq.III.2)}$$

Intuitively speaking, this definition defines a partition in which no element in C_1 proceeds any element of C_0. In other words, some elements belonging to C_1 (by Eq.III.1) may now be placed in C_0, but, no element belonging to C_0 C_1 (by Eq.III.1) can be placed in C_1 (Eq.III.2). In doing so, the subset C_0 will be enlarged, and the region shape of C_0 will be smoother. At the same time, the enlargement of C_0 is controlled to such an extent that the loss of classification accuracy is limited to a reasonably low level.

Another scheme, which is symmetric to the Eq.III.2, can be defined similarly for the following condition.

$$\neg a \in C_0,\ a \prec b, \text{ for any } b \in C_1. \text{ (Eq.III.3)}$$

In this case, we ensure that no element in C_0 is proceeded by any element in C_1.

It should be pointed out that there is a tradeoff between the enlargement of C_0 and the space requirement for the region shape representation. It is possible (and indeed quite often) that there are some isolated scattered points that belong to C_0. But, in order to exclude those isolated points from C_1, we have to sacrifice many elements from C_1. This will result in a very large C_0, a degraded classification. An approximation to the scheme of Eq.III.2 is introduced to rectify this problem. A small number of elements in C_0 are allowed to be "misclassified" into C_1. Those misclassified elements can be separated later in the intra-region coding stage that generally follows natural processing orders available for further classification of the partitioned elements.

When such region [redacted]rchy formation schemes are applied to im[redacted]primitive planes, they serve as preordering processes that set up a partial ordering among the primitives. Then, the problem of coding this ordering becomes a geometry problem for coding the shapes of the partitioned regions. Transformations such as wavelet are known to use a joint spatial/frequency domain methodology. The region hierarchy formation scheme mentioned above has an intuitive geometric explanation for this class. Transformations such as DCT and Fourier transform are purely frequency domain methods. It is interesting to note that when the above partitioning scheme is applied to a 2D DCT transformation plane, the generated regions are roughly circularly shaped bands. The popular zigzag sequence used in JPEG turns out to be a simple and reasonable approximation of these bands. Therefore, the region based coding strategy accommodates naturally both spatial/frequency domains and pure frequency domain transformations.

Automatic Region Formation in the Wavelet Transform Domain

DAC has developed an automatic region formation scheme that is used to categorize wavelet coefficients in the transform domain according to magnitude. The procedure will be outlined in the following sections. Also included are many of the design details used to develop the first version of the general region classification scheme. The advantages and disadvantages of the technique as well as performance and overhead issues will be discussed. Finally conclusions will be drawn to summarize the results.

Detecting the Regions

The wavelet transform is used to decompose the original image information into a multi-resolution hierarchy of data that is more suitable for compression. The result of completing one pass of the 2D wavelet transform is one low pass part (LL_1) and three high pass parts (HL_1, LH_1 and HH_1.) The transformation procedure is repeated using the low pass LL part as the starting point for each succeeding pass. The LL part is an approximation of the original image at a lower resolution. At each resolution level the HL parts carry the vertical, the LH parts carry the horizontal and the HH parts carry the diagonal edge information. The level of detail information contained in any particular orientation is specific to the local response generated in the choice of

wavelet kernel. T[redacted]riginal image is transformed into a uniqu[redacted]nd pass response hierarchy of detail information. In any compression process, it is the highest energy coefficients that are considered first for making a contribution to the reconstructed image. It is this transformation property that is used to partition the coefficients into regions of interest corresponding to high energy locations in the original image space.

The magnitude of the coefficients in a particular subband is related to the response that the original image information has to the transform kernel. Sharp edges in the original image (or a lower resolution LL part) are indicated by an increased kernel response in terms of coefficient magnitude, in the general vicinity of a specified area of concern. Other image changes that are not as abrupt will translate into a gradual response that is less significant. In those areas where little or no change occurs, the wavelet transform will indicate little or no response.

Level 1 Kernel Response Threshold Images

DAC's automatic region formation scheme makes use of coefficient magnitude information to categorize the data. The starting point for the procedure is at the highest resolution level of the wavelet transform hierarchy looking at the three detail data sets (HL_1, LH_1, HH_1.) A 2 bit representation of each detail orientation for a 256 by 256 Lena image is given in Figure 9. These images are obtained by using threshold values to categorize the coefficients by magnitude. The wavelet kernel used in this case is the standard 9-7 filter set implemented in a lifting scheme.

A decaying histogram procedure is used to determine the threshold values in each case. In this particular analysis the coefficients are partitioned into regions according to decreasing levels of magnitude; 10% of coefficients are partitioned into region 1, 15% into region 2, 25% into region 3 and 50% into region 4. Note that the largest coefficients appear darker in the images while the smallest appear lighter. The threshold values are calculated automatically for each orientation. The values used to generate the partition in each case are given in Eq.III.4 (assume C_i is the magnitude of the coefficient under consideration.)

HL_1: $0 \le C_i < 2$, $2 \le C_i < 4$, $4 \le C_i < 8$ and $C_i \ge 8$
LH_1: $0 \le C_i < 3$, $3 \le C_i < 5$, $5 \le C_i < 14$ and $C_i \ge 14$ (Eq.III.4)
HH_1: $0 \le C_i < 2$, $2 \le C_i < 3$, $3 \le C_i < 5$ and $C_i \ge 5$

The set of threshold images appearing in Figure 9 is motivational in determining a region formation scheme based on this approach. It is apparent that wavelet coefficients can be classified in this manner to form a mask. However, the large overhead required to code each individual mask must be solved in order to make this type of classification scheme viable for implementation.

Formation of the Raw Common Mask

The first step taken in the development of a technique designed to reduce the mask overhead is to consider a common mask approach that can be used to capture the most important coefficients between all three orientations at the lowest resolution level. Given that the data range is different in each orientation, the data is normalized to the largest range in an absolute value sense. This step is taken in order to compensate for the range differences that exist between the data sets, and to ensure that corresponding pixels from each orientation can make an equal contribution in the formation of the common mask. This step is illustrated in Eq.III.5.

$$HL'_1 = NewRange(HL_1) = ScaleRange(MaxRange(LH_1, HL_1, HH_1))$$
$$LH'_1 = NewRange(LH_1) = ScaleRange(MaxRange(LH_1, HL_1, HH_1)) \quad \text{(Eq.III.5)}$$
$$HH'_1 = NewRange(HH_1) = ScaleRange(MaxRange(LH_1, HL_1, HH_1))$$

Once the data ranges for the smallest two sets have been scaled to that of the largest, the common mask values can be determined. The next step is to generate a new normalized data set by taking the maximum value between scaled orientations at each pixel location. Let $C'(i)$ be the new coefficient obtained in this procedure and let C'_{HL1}, C'_{LH1} and C'_{HH1} be the individual scaled values from each orientation. The new data set is formed using the step illustrated in Eq.III.6.

$$C'(i) = MAX(ABS(C'_{HL1}(i)), ABS(C'_{HL1}(i)), ABS(C'_{HL1}(i))) \quad \text{(Eq.III.6)}$$

The final step in f[redacted]ng the common mask is to threshold the [redacted]coefficients based on the histogram decay and the percentage of the total data amount to encapsulate in each region category. The same values mentioned earlier are used in this case (10%, 15%, 25% and 50%.) The raw common mask categorization image appears in Figure 10. The threshold values used to generate the mask are given in Eq.III.7.

$$H'_1: \ 0 \le C_i < 5,\ 5 \le C_i < 11,\ 11 \le C_i < 26 \text{ and } C_i \ge 26 \qquad \text{(Eq.III.7)}$$

Note the level of detail contained in this image. High frequency edge information representing the most important coefficients appears in the darkest areas. Low frequency changes representing areas of the image that are relatively uniform appear in the lightest areas. The combined kernel response from each orientation follows the high energy areas that exist in the original image.

Common Mask Conditioning Using the DCT Method

Generally, the raw size overhead of the common mask obtained in the previous section is still too large to be included in the final bit stream for most image processing needs (i.e. an overhead of 2 Bpp for this particular 4 region mask). This section outlines the next step in the automatic region formation scheme. The DCT transform is used to reduce the overhead size to an acceptable amount.

The DCT is not applied directly to the common mask data in its mult-valued form. Rather it starts by considering each region contained in the common mask as an binary data set such that the sum of the individual data sets is equal to the multi-valued mask in functional form. The DCT transform applies the transform of the sum of functions in the same manner as taking the transform of each individual function and summing the results. Furthermore there are unique spectra in each case that should be considered separately for a fully integrated analysis. The individual region masks obtained by decomposing the raw common mask for the Lena image appear in Figure 11. Notice that although there are only three images, the fourth region channel (the background) is implicit.

The 2D fast DCT algorithm is used to transform each binary data set into the frequency domain for spectral analysis. The spectra for the upper three regions

appear in Figure [redacted] Please note that a log scaling of the coef[redacted]nts is used in the images to exaggerate the appearance for display purposes. Notice how the low energy content is dispersed through the spectrum in each case. Even at such low energy levels, the energy is localized in some areas.

It is apparent from the spectral results of Figure 12 that the magnitude of the DCT coefficients decreases rapidly in moving down each spectrum. The interaction of the content retained for each partition is used to guide the classification procedure. The interrelationships that exist in each binary spectrum must be investigated further.

A plot of the overall summed spectra in short integer format appears in Figure 13. The conventional zigzag ordering is used to re-group the coefficients before quantization. Note that only the first half of the data in reflected in the plot. The outer 8K are omitted since they do not change much from the DC level.

The first consideration observed in the plot is that the coefficient magnitude deteriorates at an alarming rate over a relatively small number of coefficients in the first portion of the spectrum. The second observation is that most of the spectral energy is concentrated in the first 12% of the data. Together these observations suggest that implementing a low pass spectral filtering stage, followed by careful quantization will reduce the amount of data that must be retained to capture the region formation concepts at the lowest level (in this common mask partitioning scheme). Note the apparently large dc offset. Part of this level is introduced by scaling the original wavelet data sets to a unified range. The rest of the offset is due to the DCT transform.

Low Pass Filtering the Common Mask Spectrum

The total mask spectrum must be filtered carefully to reduce the excess content that has little contribution to the region trends that exist in the retained coefficients. The main concern is to determine how to quantify the filter size such that a dynamic implementation may be obtained that will work accommodate any arbitrary image sizes.

The first step taken towards finding a dynamic solution to this problem comes from experimentation. In looking at the result of using zigzag ordering, coefficient

magnitude and significance deteriorate at some exponential rate. Experimental evidence suggests that as the size of the original image increases, the number of coefficients that must be retained to construct a similar quality mask increase on a logarithmic scale. This is an important observation since doubling the original image dimensions translates into a log 2 increase in the number of retained spectral coefficients.

Assume that β is a general log base to begin our discussion. If the filter size for an N by N common mask spectrum has been determined experimentally as ϕ_N, then what is the size of the filter for a 2N by 2N mask spectrum? If the unknown filter size as ϕ_{2N}, then the following simple logarithmic relationship can be used to compute the new filter size.

$$\frac{128}{256} = \mathrm{Log}_{\beta}(\phi_N / \phi_{2N})$$

$$\phi_{2N} = \phi_N \cdot \beta^{1/2} \qquad \text{(Eq.III.8)}$$

A number of experiments were conducted to determine a reasonable filter size that can be used as a base for filtering common mask spectra. It has been determined that for masks of size 128 by 128, a filter size of 40 to 45 diagonal rows yields reasonable results. The table in Figure 14 give the filter sizes for raw mask sizes using 1.375 for β. The logarithmic base β can be used for final calibration in the current design implementation.

Notice the filter size specified for a common mask spectrum of 32 by 32 is given as 32 diagonal rows. This value is selected to calibrate the filter sizes for the larger data sets. The results indicate the filter size required for a first cut of the common mask coefficients does not have to be exact since the distribution of the spectral content may change slightly from one image to the next. However, it must be large enough to capture the majority of the most important spectral content for each region partition.

Spectral Quantizat[redacted]Techniques for Filtered Coefficients [redacted]

DAC has developed two techniques for quantizing the filtered mask spectrum. The first approach is basically a modified uniform quantization procedure that is used to verify the concepts. It proved useful in confirming the algorithms and developing the concepts, but it lacks robustness and flexibility. The second technique is a general approach that is much more robust and elegant. It is dynamic since it tracks the magnitude of the spectral content in determining how to quantize the coefficients.

A First Quantization Approach

The first approach combines the filter dimension determination into the quantization procedure. A base mask size of 128 by 128 is use as the starting point in this approach. Assuming that a logarithmic relationship exists for coefficient importance along the diagonal order, a quantization technique can be developed to exploit the relationship.

The main premise is that there are bands of spectral coefficients that can be used to guide the quantization procedure. The importance of the coefficients in each band decreases along the spectrum. This concept is illustrated in Figure 15.

The number of diagonal zigzag rows to include in each band is a function of the filter size and the original common mask size as well as the spectral content. As a first approximation, a table look up technique is used to determine the number of rows to use in each band. The base band sizes determined experimentally to give reasonable partition and reconstruction results for a 128 by 128 mask are given in the table in Figure 16.

Notice that the number of diagonal rows doubles in each step. The number of bits selected to quantize each band decreases from one band to the next in the following fashion.

- Band 1: 16 bits
- Band 2: 10 bits
- Band 3: 8 bits
- Band 4: 8 bits

A series of band s[redacted]were found experimentally of mask dime[redacted]s by powers of 2 increments. The results are in the table of Figure 17. These values are used to partition the spectral content for quantization. This information together with the number of bits used in each band and the filter size measurement determines the common mask overhead of an image.

The mask overhead based for this quantization scheme is tabulated in the table in Figure 18 for the corresponding image sizes. There is an additional overhead header included in this result for mask reconstruction on the decoder side. Note that the mask overhead is the same size for both grayscale and color images since it is generated from the intensity information. However, the percentage overhead in the color case is reduced by a factor of 3 since it can be distributed over 3 channels.

After applying the inverse DCT on the quantized coefficients a low pass approximation of the original common mask is obtained. It is this result that is used to guide the region channels in classifying and processing the wavelet coefficients. The result indicated in Figure 19 is obtained by applying this technique to the first wavelet transformed level for a 256 by 256 Lena image. From Table 17, the overhead for this mask is 1021 bytes.

Notice the high energy changes have been eliminated by low pass filtering (especially the changes that occur over small portions of the image). However, the mask does succeed in capturing most of the essential parts of the original image. This mask is common to all orientation at this level and used to partition the coefficients into region categories.

A Second Quantization Approach

There are some apparent problems with the underlying procedures used for quantization in the first approach. The first of these is that the coefficient band classification is fairly rigid and not that flexible. The filtering and quantization stages are tied together and fully independent. Affecting one parameter has consequences to the other. It is true that a logarithmic relationship exists in the spectrum that can be exploited to determine a reasonable filter size, but the quantization step must be tailored from the filtering stage.

DAC is currently implementing another quantization scheme for mask filtering that is much more dynamic and flexible. Quantization is no longer tied to the filtering stage and the coefficient banding technique of the previous approach. The current technique follows the actual content of the spectrum in forming quantization categories. In this way, coefficients are grouped into categories according to the energy gradient of the spectral distribution. The concept is illustrated in Figure 20.

The rate of spectral decent together with position information within the spectrum can be used to quantize the number of bits used to categorize the data. This process can be controlled precisely. The procedure can be calibrated as required by studying the effects of changing image dimension in terms of an optimal quantization change.

The previous quantization technique can be used to estimate the mask overhead for the new scheme proposed here. Consider the same base conditions presented for the banding approach applied to the 256 by 256 Lena image. The spectral filter size in this case is 44 diagonal rows. If 4 quantization intervals are used at 16, 12, 8 and 4 bits each the mask overhead can be estimated. Let N_i be the number of coefficients in each interval, b_i the number of bits used to quantize this interval, and O_M be the total mask overhead. A rough estimate of the new packed size is calculated in Eq.III.9.

$$O_M = \sum_{i=0}^{3} N_i . b_i$$
$$O_M = 6 \times 16 + 39 \times 12 + 186 \times 8 + 804 \times 4 \qquad \text{(Eq.III.9)}$$
$$O_M = 5268 \text{ bits} \quad (\sim 659 \text{ bytes})$$

Even though the actual overhead has not yet been determined, from the insight gained in using the first quantization approach a saving of 30% is not unreasonable. This promises to be a substantial saving in the mask overhead. There are ways to utilize saved overhead. The first is to reduce the amount of mask information in the code stream. This approach will retain the same mask quality leaving extra space for

refinement inform[redacted] in the code stream. The second way [redacted] utilize the saved overhead is to retain a better quality mask in keeping the overhead size constant.

The Common Mask and Multi-resolution Classification

A translation technique is required to apply the region mask at other resolution levels. The simplest approach is to look at the significance of the mask coefficients in a small block to determine an appropriate value for the next resolution level. This process is repeated until a mask for each resolution is obtained. The most logical approach is to take the largest coefficient through to the next level.

$$Mv_{(L+1)i,j} = Max(Mv_{(L)2i,2j}, Mv_{(L)2i+1,2j}, Mv_{(L)2i,2j+1}, Mv_{(L)2i+1,2j+1}) \qquad (Eq.III.10)$$

Some heuristic approaches have been investigated for resolution translation. One approach is to make decisions based on the sum of the individual mask coefficients. The upper level masks are affected by either enlarging or shrinking the encompassed coverage regions. However, there is currently no conclusive evidence confirming the validity of this approach and more experimentation is required.

It is worth mentioning at this point, that DAC has begun investigating the wavelet kernel ROI mask formation technique of verification model (VM) 3.0 as a method of resolution translation for other transformation levels. Preliminary investigations indicate this technique introduces a gradient bias for regions of higher classification level in the translation. A combined hybrid approach may have some advantages.

The rest of the common mask hierarchy obtained by using the simple 4 to 1 resolution translation techniques and observing the most important region as key is given in Figure 21. The complete set of region masks is used to classify regions of importance in the wavelet transform domain according to coefficient magnitude.

Misclassification and the Common Region Mask

It should be noted that while it is true that misclassification occurs in this technique, there is a tradeoff between code stream overhead and region channel accuracy. The original raw common mask size is 32 kBits for a 4 region mask. The basic uniform

quantization sche[redacted]compresses this amount to 8 kBits. [redacted]modified scheme suggests that mask compression of 5 or 6 to 1 can be achieved with a tradeoff between mask accuracy and code stream overhead.

The low pass filtering and spectral quantization steps can be tailored to deliver similar quality masks for most images in a dynamic implementation. In addition the partition can be studied in regards to how the original grouping is affected by the filtering step. For the Lena image example the original partition is 10/15/25/50%. After filtering the coverage ratios used for bit budget calculation and the code stream distribution (see multiplexer in Ch.V.) changed to approximately 9/31/33/27%. There may be some advantages in adjusting the original partition such that the data coverage ratios are controllable in the final stages.

The question of miscoverage must be addressed in all lossy region formation techniques. It is the subsequent modules that are affected by the misclassified information. Thus the interaction between the DCT region formation module and the subsequent sorting and packing modules must be understood in any attempt to obtain the best region channel results for this classification scheme.

Handling Misclassification in a Region Formation Scheme

Subsequent stages in the compression algorithm are already tailored to process information in a descending level of importance. Most wavelet implicit quantization procedures process the coefficients in bit plane order of importance. What this means in terms of any region formation scheme is that the information encompassed in each region is further partitioned by bit level processing. In the common mask approach, further processing causes the most important misclassified coefficients to filter to the top at a faster rate than misclassified coefficients of lower importance. Effectively bit level processing can be considered as a second partitioning stage that deals with misclassification.

There is a tradeoff between the accuracy of the original region partition and the additional bit level processing overhead placed on subsequent sorting stages. It is the final processing stages that implicitly accomplish the final coefficient partitioning. DAC is currently using two bit plane partitioning core technologies for both grayscale

and color imag[redacted]nd binary partitioning for bi-level i[redacted]s is now under implementation.

Bit Plane Sorting for Common Mask Miscoverage

The concept of bit plane sorting is a well know technique used to organize priority in a distributed list of data. 1D sorting a list of information generates a map specific to that list in an optimal fashion. The largest values are mapped first as the list transposition routine progresses. The order of decent is the standard order used to classify information if considered from the normal processing standpoint where no regions are involved at all. As a result, most of the misclassified information is considered for further classification and are thus biased to be included first in the final code stream order. The only difference is that now there is a region description overhead as well as the final mapping overhead of sorting the region of interest partitions in a hierarchical fashion.

Currently DAC has not documented the exact overhead introduced by region processing in comparison to normal processing modes that do not use regions. But the final results obtained by using the common mask approach are encouraging to say the least. At this level the underlying routines do not care where the information to be sorted originated. The ordering technique will run to completion whether region channels are involved or not. The original misclassified coefficients generated in the first cut will filter through to the final ordering set by the sorting stage. There is an important tradeoff here between the sorting cost and the region overhead that must be considered in the final design implementation.

Bit Plane Ordering for Common Mask Miscoverage

DAC has developed a quad tree ordering technique (see EQW Ch.4) to track coefficient importance in both normal and region based processing modes of operation. The normal quad tree approach is to track the leading one position through each block under consideration in a recursive fashion. A map is generated in the process for each entry based on importance such that decoding follows the same order. At this basic level the quad tree order is an efficient method of processing the multi-resolution decomposition image information in an effective manner.

The tracking or positioning overhead introduced by mapping data is generally smaller than in the 1D sorting case. Retaining vertical and horizontal positioning information is highly advantageous (in most image processing applications) since the underlying technology is generally consistent in this medium. The 1D sorting approach sacrifices the natural vertical correlation that may exit (although it can be partially recovered by bit level entropy coding in the sorting routine.) However each of the two sorting techniques can be used from an operational standpoint to produce a similar net effect. One may produce slightly better results than the next, but the control architecture is consistent. Each final ordering routine can be tuned to meet specific processing needs. The 1D sorting approach may be a useful candidate for bi-level image processing, and DAC is currently investigating this possibility.

The quad tree approach has currently been modified to operate in arbitrary region processing modes. The 1D sorting routine tracks the importance in a list of information specific to the block under consideration. That is the region partition is distributed in 1D hierarchical lists of information. The original preordering of the coefficients is captured in the common mask procedure. Thus the decoder map already exists for the inverse ordering.

Initially DAC's region processing quad tree begins by building an information map for each region channel to guide the classification. The core engine has been modified to drops blocks not contributing to the final ordering in each region category. Currently the overhead comparisons for each sorting case have not been documented. The results will be available from DAC once the region based quad tree design implementation is finalized (currently under implementation at DAC.)

In each case, the misclassified coefficients obtained from the original preorder mask partition are considered again for ordering in the sorting stages that follow. Effectively, the most important misclassified coefficients still have the opportunity to make important contributions to the final code stream as well as improving the reconstruction quality measurements in the decoded image. The difference between this approach and most efficient classification approaches is that the original processing order used of the efficient scheme is somewhat distributed through each region channels. However the misclassification in each region channel is forgiven to a certain extend the subsequent bit level sorting module. One of the prices that must

be paid for any automatic region classification scheme is that optimal processing orders that exist in standard approaches must be considered in a different context. The original ordering concepts used for standard procedures must be extended to include other ordering techniques specifically tailored for operation in arbitrary region processing.

General Region Processing Concerns

There is one important hurdle that must be overcome in the design of any arbitrary region processing technology. There are times when rigid tile or block control is the optimal way to process image information. However, it is difficult to achieve arbitrary accuracy in a strict processing order. DAC's 1D and 2D sorting techniques address this issue in design implementation. In our opinion arbitrary region processing channels can be defined in a still image environment. Careful study and analysis suggests that this is a viable approach not only for the DCT common mask approach but also for other arbitrary region formations techniques that have not yet been developed. Furthermore region classification may have interesting properties that can transferred to video processing environment where the region of interest concept and the object motion vectors can be considered together in a unified approach to region of interest processing.

DAC's region technology in its original form was designed to include region processing capabilities from the start. All internal modules can be operated in both normal and region based modes of operation. The MUX architecture (introduced in Chapter V) is designed to meet operational requirements in both normal and region processing modes. It is possible to extend the notion of block based image processing to a level where a block of information is interpreted in an arbitrary way that depends only on the underlying core processing technology. In this way, blocks can be considered as individual processing units each fully capable of making a contribution to the final code stream. Furthermore control architecture can be developed to fully exploit the region processing capability. All region formation schemes, whether arbitrarily defined user regions of interest, automatic region formation schemes or any other specially designed region classification technique, can be processed in a common framework. The same overall processing architecture can be tailored to

operate in an optimal fashion for both normal and region processing modes of operation.

Additional Processing Concerns

One of the important concerns in the JPEG-2000 community regarding the development of a new compression standard is that it must be able to support ROI processing. There are a number of techniques that have been tested with some success. Some of these techniques are listed below.

- ***Sequence based mode.*** When this mode of operation is used, each set of data encapsulated by a specific region is treated as a separate sequence. The sequences are coded independently.
- ***Scaling based mode.*** In this mode of operation, the magnitude of each ROI coefficient is increased using a predetermined shift factor. If multiple ROIs are used with increasing levels of importance, multiple shift factors distinguish each ROI. The effect of this scaling technique is to force the ROI coefficients to be encoded first.
- ***ROI from the middle.*** In this mode of operation, a concern of coding an ROI in the middle of the bit stream is addressed. This is useful for progressive transmission of images where the user is given the opportunity to focus on a specific ROI before the complete set of image data arrives at the decoder. This is important for medical image processing.
- ***ROI without sending mask information.*** This mode of operation is a special case for scaling based mode. The coefficients for each region are scaled such that the magnitude of each ROI exceeds that of the ROI of less importance. The decoder knows which ROI the coefficients belong to based on the magnitude.

The ROI coding technique that DAC has developed generally falls into the sequence based category. But internal modules have been tailored to operate in both block based and arbitrary block sized processing units. DAC is currently studying the different mask and region formation techniques to determine the effect each ordering and partitioning approach has on the overall organizational structure required for region channel processing. The current implementation includes a scalable ROI mode of operation implemented without actually shifting the ROI coefficients encompassed

by the classifica[redacted] Data scaling is handled in the MU[redacted]ntrol architecture. Internally, the scaled version is a special case that can be implemented in MUX control.

Preliminary study has begun for the last of the four categories cited above. ROI from the middle implementations can be addressed by using resolution a progressive processing order. This type of ordering is supported in the MUX architecture. DAC is currently considering a design implementation for client side region of interest requests in a resolution progressive architecture for client-server region interaction.

ROI without mask information makes sense in theory. However it places a huge tax on the subsequent sorting and classification stages. Generally speaking, the overhead cost introduced by additional sorting is comparable to the size of a bitmap mask that could have been used to classify the data initially. The shift introduced to each ROI must be processed in the sorting stage. Eventually sorting must traverse from top to bottom. Coefficient scaling under tight control has some definite benefits in many cases. However it would be highly inefficient in a general region processing architecture because of the high sorting costs as a result of traversing all region channels. In a region formation scheme of 4 or more regions, the total bit level difference between all ROI is too large to process entirely. On the other hand, if the resolution of the overall partition is reduced (less bit levels), less room is available to form an accurate region classification. There is some common ground that must be explored in order to determine the optimal processing mix.

Intra-Region Coding

The segregation of image primitives into regions provides a basis for accessing the image contents. Compact representation of image primitives is achieved via intra-region coding.

Independent coding units (ICUs) are the building blocks for intra-region coding. In RICS, intra-region coding with ICUs is designed with a set of objectives.

(1) *Full data independence*. From the notion of ICU, the encoding and decoding of a region is done without referring to the data of any other regions.

(2) *Allowance for multiple coding schemes.* Each ICU may be coded using different coding schemes. New coding schemes can be added to the system (modular openness.)

(3) *High coding efficiency.* Depending on the characteristics of the primitives in a given ICU, one or more efficient coding schemes can be selected to produce a code block with high compactness.

(4) *JPEG and JBIG transcodability.* It is preferred to have a single JPEG 2000 coding platform that also accommodates JPEG and JBIG modes of coding.

(5) *Scalability of code stream.*

(6) *Error resilience.* The ICUs are natural blocks for bit error localization.

(7) *Parallelism.* There is no data dependence between ICUs and so the encoding and decoding of all ICUs can be performed in parallel.

Intra-region coding in RICS involves the following steps.

(1) Choosing a coding scheme.

(2) Determining ICU structure.

(3) Coding.

(4) Pre-packing.

In the rest of this section the first three steps are described.

Choosing a Coding Scheme

For coding an ICU, the current RICS design employs six categories of coding schemes.

- Zigzag DCT Coefficients Packing (JPEG baseline scheme)
- Embedded Quad tree (or similar schemes)
- Embedded Zero tree (or similar schemes)
- 1D Progressive Sorting Schemes
- Block Based Quantization
- JBIG Coding Algorithms

There is no general restriction on what schemes have to be used for coding a given ICU. However, certain preferred embodiments may apply. For example, if DCT is used at the transform step, then the zigzag DCT coefficient packing might be preferred (but not restricted to).

Choosing ICU Structures

While being a logic concept, an ICU still has a geometric structure. The ICU structure is directly relevant to the coding scheme chosen. Currently, the RICS recognizes three types of ICU structures (Figure 22).

Type 1 ICU Structure: 8x8 Blocks

This type of ICU is designed exclusively for use in JPEG mode, although it is not necessary that DCT transform coefficients be coded this way.

If there is no region defined in the image and the JPEG mode is chosen, the entire image is paved with type-1 ICUs.

If a region $\boldsymbol{R}$ is to be coded using JPEG mode, the set of ICUs that covers $\boldsymbol{R}$ form a unique, minimal pavement for $\boldsymbol{R}$, in the following procedure.

Step 1. Let $y_{\min}$ be the first row from the top that has at least one pixel in $\boldsymbol{R}$, $y_{\max}$ the last row, $x_{\min}$ the first column from the left, and $x_{\max}$ the last column.

Step 2. Let $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$ be respectively the top-left and the bottom-right corners of the bounding rectangle of $\boldsymbol{R}$.

Step 3. Starting from the top left corner, pave the bounding box completely with type-1 ICUs.

Step 4. Remove those ICUs that cover no pixels in $\boldsymbol{R}$.

After this procedure, the remaining ICUs form the minimal pavement of $\boldsymbol{R}$ (Figure 22).

Type 2 ICU Structure: Rectangles

Type-2 ICUs are rectangles. Except for the JPEG and embedded zero tree schemes, any other intra-region coding scheme uses type-2 ICUs to pave and code a region. There is no general restriction on the dimensions of a type-2 ICU. It is usually

defined by the selected coding scheme. For example, both embedded quad tree and explicit block based quantization can use arbitrary sized rectangles, with some preferred embodiments (such as the 64x64 blocks used in EBCOT of VM 3.0 (A)). It should be noted that once a region is paved by type-2 ICUs, different coding schemes can be used for each ICU.

For subband decompositions, no type-2 ICU is allowed to cross subband borders. Essentially, this means that type-2 ICUs support only various types of intra-band coding methods.

Type-3 ICU Structure: Pyramids

The third type of ICU is the pyramid structure, which is used by various inter-band coding methods, such as the embedded zero tree. In the current RICS design, there is one limitation on the region definition procedure for type-3 ICUs: a region must be specified in the top-down manner, from lower resolution to higher resolution in the decomposition pyramid. That is, for a given region definition $\boldsymbol{R}$ in LL, every element in $\boldsymbol{R}$ defines a set of type-3 ICUs (i.e. in three spatial orientations, respectively).

Let $C^{LL}(i,j)$ be a wavelet transform coefficient at spatial location (i,j) in the LL subband, $C_l^r(m,n)$ be a wavelet coefficient at spatial location (m,n) in the subband at level l $(l=1,2,...)$ in orientation r $(r=1,2,3)$. Assuming that a 5-level wavelet - decomposition is used, the three type-3 ICUs defined by $C^{LL}(i,j)$ can be represented in the following fashion.

In orientation 1:

$$\{C_5^1(i,j)\} \cup \{C_4^1(2i,2j),\ C_4^1(2i+1,2j),\ C_4^1(2i,2j+1),\ C_4^1(2i+1,2j+1)\} \cup \ldots$$

In orientation 2:

$$\{C_5^2(i,j)\} \cup \{C_4^2(2i,2j),\ C_4^2(2i+1,2j),\ C_4^2(2i,2j+1),\ C_4^2(2i+1,2j+1)\} \cup \ldots$$

In orientation 3:

$$\{C_5^3(i,j)\} \cup \{C_4^3(2i,2j),\ C_4^3(2i+1,2j),\ C_4^3(2i,2j+1),\ C_4^3(2i+1,2j+1)\} \cup \ldots$$

Notice that the primitive $C^{LL}(i,j)$ is not included in any the three ICUs. Instead, the LL subband is coded differently, using the type-2 ICUs.

Coding Type-1 ICUs

Type-1 ICUs are defined exclusively for coding 8x8 DCT coefficient blocks. Coding of type-1 ICUs follows the JPEG baseline algorithms.

Coding Type-2 ICUs

Once a region is decomposed into a set of non-overlapping type-2 ICUs, each ICU is coded independently, and the collection of codes for all ICUs is the coded representation of that region. In this subsection we describe three techniques that may be used for coding type-2 ICUs.

Using Embedded Quad tree

Embedded Quad tree Wavelet (EQW), Figure 24 is an efficient and fast method for type-2 ICU coding. This technique implements an embedded progressive sorting scheme in a quad tree-like structure. In contrast to inter-band coding methods, the EQW explores the intra-band self-similarity of the wavelet decomposition coefficients. The EQW-produced code-stream realizes scalability by pixel-precision (the scalability by spatial resolution is realized by the multiplexer.)

Depth-First and Breadth-First Quad trees

The EQW method can be used for both lossless and lossy coding. In lossless coding, the primitives to be encoded in the ICUs are coefficients of a reversible wavelet transform. In lossy coding, after the wavelet transform, each transform coefficient is represented in a fixed-point binary format - typically with less than 16 bits - and is treated as an integer.

In each ICU, the maximum coefficient magnitude M is determined. Then a value N is found which satisfies the condition $2^N \le M < 2^{N+1}$. The EQW works in a bit-plane manner: the initial bit-plane is set to 2^N, followed by $2^{N-1}, 2^{N-2} \ldots$ and so on. A binary significance map is produced for every bit-plane by comparing coefficients in power of 2 increments.

Figure IV.3 illustrates the EQW encoding process on a single bit-plane. Two working lists are used. One is called the list of significant primitives (LSP). Each entry in LSP corresponds to an individual primitive in the ICU. The LSP is initialized as an

empty list. Another list is called the list of insignificant blocks (LIB). Each entry in the LIB is registered with the coordinates of the top-left corner of the block, together with the width and height information. Each LIB entry represents an individual primitive of width and height equal to 1. At the highest bit-plane 2^N, the ICU to be encoded is put into the LIB. A temporary list of insignificant blocks (TLIB) is used for refreshing the LIB after each bit-plane coding pass is completed.

There are two methods used to add the four sub-blocks into the LIB that occur as a result of the quad tree decomposition. The first method, known as depth-first quad tree coding, is to insert the four sub-blocks into LIB at the position of their parent block. In this case, the four child blocks are evaluated immediately after the parent block. This rule is applied recursively until no more subdivision is possible. This means that control returns to the next entry in the LIB only after the present entry is completely encoded up to the highest resolution level. The second method, known as breadth-first quad tree coding, is to add the four sub-blocks under consideration to the end of LIB. In using the breadth-first process, all parent blocks at the same level will be processed first, followed by their respective children blocks.

After all entries in the LIB have been processed, the entries in TLIB can be reordered according to certain pre-defined set of rules (this is an optional step.) Experimental evidence suggests that a higher PSNR can be achieved by using an effective reordering scheme. Then the LIB is replaced with the TLIB and the coding resumes at the next bit-plane.

The next step is the refinement pass. The N^{th} bit is output for entries in the LSP. Then, the process resumes using the new LIB and a new bit-plane set to **Error! Objects cannot be created from editing field codes.**.

EQW with VLC

In order to use variable length coding (VLC), the standard EQW algorithm of the previous section is modified as follows.

Step 1. **Initialization:** output n satisfying the inequality $2^n \le \max\{|C_{ij}|\} < 2^{n+1}$, set the list of significant primitives (LSP) as an empty list. Put the ICU to be coded into the LIB.

Step 2. **Bit-Plane Sorting**: for each entry of the LIB, perform quad tree coding. If all primitives within the block are insignificant, output a 0 bit and add this block to the temporary LIB (TLIB).

Otherwise, look at the significance of the four sub-blocks. Then according to the VLC table, output the corresponding VLC sequence (see the table in Figure 25). If significant, sub-blocks that are not a single primitive are inserted into the LIB at their parent position. If insignificant, they are added to the TLIB. If the sub-blocks are single primitives, they are added to the LSP. Sign bits are output only if units are significant. If insignificant, the units are added to the TLIB.

Step 3. **Reordering** (optional).

Step 4. **Refinement:** for each entry in the LSP, except those included in the last sorting pass, (i.e. with same n), output the nth most significant bit of $|C_{ij}|$.

Step 5. **Quantization Update**: decrement n by 1 and go to step 2.

Some other researchers have also proposed the embedded coding methods where quad tree is used to explore the intra-band similarity [1][2]. However, some major differences exist in how the quad tree operates. The work of [2] has suggested that empirically the quad tree decomposition should stop at size 16x16. However (as a result of our experience and experimentation) this size may not always be the optimal choice. Even without VLC, the efficiency of DAC's EQW is comparable with EZW. In fact, for a 2x2 block, if it is insignificant, one zero can represent four zeros. However, when a 2x2 block is significant, quad tree coding is inefficient.

Using Block-Based Quantization

The embedded quad tree (EQW, EQPC, etc.) is a special case of the more general block-based quantization techniques. Other block-based quantization algorithms, such as EBCOT, can also be very efficient for type-2 ICU coding.

Using JBIG Algorithms

[1] F. Kossentini et al. "Embedded Quadtree Predictive Codec", ISO/IEC/JTC1/SC29/WG1/N667.

It is noted that in [redacted]plicit quantization schemes using progr[redacted]e bit-plane coding methods, including the embedded zero tree and the embedded quad tree approaches, at each bit-plane, the ICU coder is actually dealing with a bi-level image (the significance map.) Therefore, by adopting those efficient JBIG algorithms as ICU coders in a JPEG 2000 system, one can realize two modes of interplay with JBIG. In the following EQW is used as an example to illustrate this idea.

In the first mode, the ***transcode*** mode, the JBIG ICU coder and EQW ICU coder provide a two-way transcode between JPEG 2000 and JBIG (Figure 26). In this transcode, the EQW procedure is called for bi-level type-2 ICUs, with the total number of bit-planes being set to one. Note that the sign bit coding in the EQW algorithm should be skipped.

In the second mode, the ***embedding*** mode (Figure 27), the proper JBIG routines are called as the bit-plane coder in the EQW routine for certain bit-planes and for certain decomposition sizes (the quad tree decomposition stops at this particular size and the coding is handed over to the JBIG routine). Since the algorithms for bi-level data coding are also evolving with the compression technology, having an embedding JBIG mode reserved in the JPEG 2000 system can make the Standard evolve with its sister technologies.

Coding Type-3 ICUs

Type-3 ICUs are defined to support various inter-band coding methods for wavelet transform coefficients. In particular, the methods of EZW, SPIHT, etc., are known to be efficient approaches for coding the type-3 ICUs. In testing these approaches, it is noted that the standard versions must be modified slightly to make them fit into the RICS architecture. Normally, these algorithms implement the ICU coding and multiplexing steps into an integrated procedure. A simple modification is required to bridge the existing architecture to the RICS architecture. The two steps must be separated, which is straightforward. Since the multiplexing stage in RICS can effectively simulate the performance of zerotree-based schemes (refer to Ch.V), this particular split in functionality has virtually no impact on the attainable coding

[2] Taubman, D. et al. "EBCOT: Embedded Block coding with optimized truncation", ISO/IEC/ JTC1/SC29/ WG1/ N1020R.

efficiency for sta[redacted] schemes. Instead it adds more flexibili[redacted]d openness to the existing architecture.

Intra-Region Coding with Different Types of ICUs

Generally there is a need for using different types of ICUs for a intra-region coding. For example, in coding the wavelet transform coefficients, the LL subband is usually encoded differently than the other high-pass subbands. Because different sets of primitives may possess different statistical characteristics, providing several coding scheme choices in the intra-region coding module may offer a higher coding efficiency. In practice, the following combinations are useful.

- In a wavelet decomposition, coding the *LL* subband with type-2 ICUs whereas the other subbands with either type-3 or type-2 ICUs.
- Tiling an image into several regions, with some of the regions paved with type-1 ICUs and others with type-2 ICUs.
- Using different coding schemes in different type-2 ICUs.

Intra-Region Coding without Any ICUs

Intra-region coding can be performed without specifying any ICUs in a particular region. In this case, the entire region is considered an ICU. If the natural geometric shape of the region fits into any of the three ICU categories, the appropriate ICU coder can be used. If the region has a very irregular shape, then the 1D coding method can be an efficient approach. Once the data for a particular region is scanned in a certain pre-defined order to form a 1D stream, the following 1D progressive sorting algorithms can be used to produce an embedded code-stream.

1D Progressive Sorting Algorithms

The EQW algorithm can be readily extended to 1D cases where a binary partition is used instead of a quad tree decomposition:

Let $L = \{c_i\}$ be the 1D list to be encoded, LSP the list of significant primitives, LIS the list of insignificant subsets, TLIS the temporary list of insignificant subsets.

Step 1. **Initialization:** Output n satisfying the inequality $2^n \le \max\{|c_i|\} < 2^{n+1}$, set the LSP as an empty list. Put the set L into LIS.

Step 2. **Bit-Plan[redacted]rting**: For each entry in the LIS, perform[redacted]inary partition. If all primitives within the subset are insignificant, output a 0 bit and add this subset to the TLIS.

Otherwise, output the two bits that reflect the significance map. For those subsets, if they are not single primitives, insert them into the LIS at their parent position if they are significant, or add them to the TLIS if they are not. If the subsets are single primitives, add them to the LSP and output the sign bit if they are significant, or add them to the TLIS if they are insignificant.

Step 3. **Reordering** (optional).

Step 4. **Refinement:** for each entry in the LSP, except those included in the last sorting pass, (i.e. with same n), output the nth most significant bit of $|c_i|$;

Step 5. **Quantization Update**: decrement n by 1 and go to step 2.

Processing with a Multiplexer Architecture

The JPEG-2000 committee has been investigating many emerging compression technologies in order to define a new still image compression standard. The emerging standard will address both present and near future image compression needs. The task of selecting what technologies to include in the new standard is not easy given the rate at which technological advances occur in this area. The primary focus of DAC has been to develop a still image compression engine that addresses the issues set forth by the standards committee.

The concept of using a multiplexer (MUX) has been adopted in some standardized processes of information coding such as MPEG. In contrast, incorporating a MUX as an integral part of a still image compression engine is rather unique, and there has been very little research conducted in this area. However it will be shown that the MUX concept is extremely useful in the development of a general purpose still image compression engine.

Background

One of the concerns in developing a new Standard is the layout design of the encoded bit stream. A protocol must be developed to guide the new standardized compression procedures. Some of the important considerations in this area are listed below.

- ***A well defined bit stream.*** All syntactic and semantic aspects of the encoded image format must be defined. The JPEG-2000 standard will encompass a wide variety of compression needs. One to one relationships will exist between fields that exist in the bit stream and functionality that exists in the core compression / decompression engine. The number of fields that exist in the bit stream is directly related to the overall functionality encompassed by the new standard.

- ***A highly degree of data accessibility.*** The bit stream must be organized in such a manner that it need not be decoded in its entirety to interpret the physical meaning of the compressed data. Highly accessible data requires that the encoded data is divided into logical units each corresponding to a specific part of the original uncompressed information. The way this was accomplished in the existing JPEG standard was to partition the original image into 8x8 blocks. Each block was compressed using the DCT. Quantization tables were developed for coding the coefficients such that the reconstructed image quality was degraded in a standard fashion. However, strict block based data accessibility may not be a suitable medium for many of the current image processing needs.

- ***Dynamic re-orderability.*** Dynamic re-orderability is a data access issue relating to the degree that encoded information can be reorganized to suite user specific needs for an image under consideration. This type of data access is an important concern for many applications, especially in the medical field.

- ***Built in error resilient.*** In any transmission medium there is always the possibility of incurring bit errors. The encoded bit stream must have a certain degree of error resilience to address this concern.

Using a MUX to organize the encoded bit stream can address these issues. The first step is to divide the encoded information into logical groups. Organizing the data in ICUs leads naturally into a high degree of accessibility. Each ICU is an independent unit. The size and position information is self-contained within each ICU. One of the

effects of using a [redacted]plexed bit stream design is that it is inher[redacted] error resilient. If any logical unit incurs an error, only that unit needs to be recovered. If the encoded data is divided into an array of logical units in preparation for MUX encoding, dynamic re-ordering is much easier to achieve.

MUX Design Overview

There are five important considerations to address in the design of the MUX for still image compression systems.

- Defining a working model.
- Defining an optimal data processing order.
- Capturing the data to be compressed in a suitable data structure.
- Defining a data priority to be used for compression.
- Packing the compressed data based on data priority and processing order.

A Working Model for the MUX Discussion

In preparation for the MUX discussion, consider that a multi-resolution decomposition hierarchy of data has been obtained for an image to be compressed. Also consider that there are three channels of data. Assume that the original image has been transformed from RGB to YUV for example. In addition to this, assume that the U and V channels have been down sampled by a factor of 4:1. This type of approach is common for lossy image compression. The data sets appear in Figure 18. The diagram illustrates a 4 level multi-resolution decomposition for the Y-channel and 3-levels for each of the U/V-channels. The wavelet transform MALLAT decomposition is used here for convenience.

This is by no means the only model that exists for the MUX design. DAC makes the distinction between region and non-region processing modes of operation in their design implementation. A hierarchy of list structures is used to control optimal bit budget distribution and flow of both non-region and region bit levels through MUX channels. The discussion presented in the next Section begins by introducing the concepts in normal processing modes. The normal MUX modes of operation are generalized later on to include specialized region and mixed processing modes.

Non-Region Data Processing Orders for the MUX

Generally speaking, given a wavelet multi-resolution decomposition of data, coefficients of a fixed magnitude contained in a lower resolution level are more important to the image reconstruction procedure than coefficients of a similar magnitude at a higher resolution level. This leads to a natural ordering of the coefficients beginning at the lowest resolution level. The natural ordering concept is an important consideration in the design of the MUX.

General Color Processing Order (Lossy Case)

Given the multi-resolution data sets of Figure V.1, a natural processing order exists for each decomposition channel. This natural order is illustrated in Figure 29. This particular order reflects a level priority inherent to each data set.

Another two orders are obtained by simply interchanging orientation data sets in a given level. The difference lies only in the convention used for ordering the detail information (i.e. LH, HL, HH). The 4-1-1 down sampling relationship that exists in the color transform domain also exists in the wavelet decomposition data since there is one less transform level for the U/V channels. This is one of the keen design considerations that can be exploited in many stages of the MUX implementation and encoding / decoding procedures. The data is organized into list structures to cover the natural processing orders that exist in the data to be compressed.

Level Priority Processing Orders (Lossy Case)

The individual orders of Figure V.2 can be combined into a single natural processing order for the example under consideration. The new order is obtained by interleaving the expressions of Figure 29 using the inherent decomposition level priority of the data. The result is illustrated in Figure 30.

Notice that in this particular order, the 4-1-1 color down sampling relationship is maintained. In terms of the reconstruction process, the complete level 4 Y-channel information (i.e. LL_4-HL_4-LH_4-HH_4) is required together with the U and V channel low pass information (i.e. LL_3 in each case). This is the information the decoder will need first to reconstruct the low pass inverse color transform image at the lowest inverse wavelet resolution level. A "level split" on the U / V data channels at the

lowest inverse wa[redacted] resolution level is used to exploit the na[redacted] relationships that exist for both inverse transform spaces. A similar order exists for the lossless case where full data sets are used.

Color Interleave Processing Order (Lossy)

Another processing order that exists for color images in the wavelet transform domain is obtained by interleaving the color channel detail information. The order is illustrated in Figure 31. This type of ordering may be suitable for certain applications such as those that require the data to be encoded for a progressive download. As in the previous case, the down sampling relationships that exist are maintained in both color / wavelet transform spaces.

Lossless Color Processing Orders

As in the lossy case, similar processing orders can be designed for lossless color image compression where full data sets exist for all color channels. In this case, corresponding orientation information from each wavelet channel is related in the original inverse color space. These data should be processed and grouped together in the wavelet transform domain and eventually the final bit stream.

The natural processing order for lossless color image processing is illustrated in Figure 32. The color interleave order for lossless image processing is illustrated in Figure 33. Note that in each case, the full decomposition data set for each channel is maintained. The inherent relationships that exist in the color transform space are maintained in the wavelet transform space for ordering and reconstruction of the original image.

Capturing the Data in a MUX List Structure

There are a number of candidate algorithms under consideration by the JPEG-2000 committee for optimal quantization of the wavelet coefficients. A common approach used to quantize the wavelet coefficients in a progressive manner is to use an optimal bit plane ordering technique. DAC has testing a number of bit plane ordering techniques in both one and two dimensional schemes. Bit plane ordering techniques are generally classified as using implicit quantization for the wavelet data sets. Coefficients are packed into the final bit stream based on bit level priority. The bit

plane processing [redacted]nology will be used to introduce the dat[redacted]ucture used in the MUX design. However, the MUX processing discussion that follows is not restricted to bit level processing algorithms.

The general ordering and processing relationships are useful for introducing the MUX control architecture at a basic level. However, the key to the operation and many benefits of the MUX technique is in the data structure design used to encompass the natural ordering concepts. The first step is to define a list structure for each level, channel, and orientation of data that exists in the wavelet transform space. A typical example of this type of structure is given in Figure 34.

A general list structure for each level data set can be utilized in many ways. A lossless "prepack" of multi-resolution hierarchy information is useful for exploiting the many relationships that exist in the data taken in MUX list processing context. The data and information contained in each list is used to control how the final bit stream is organized. All data packed into the final bit stream must conform to this structure. New fields may be added, but the basic operation of the structure remains the same. Most implicit quantization techniques are implicitly decodable. In other words, minimal header information is required for each list. The MUX architecture organizes the lists or units of information into a bit stream that is scaleable in terms of bit precision and resolution and is controllable by the many MUX modes of operation.

As an example, suppose that EQW (DAC's current two dimensional bit level processing design) is used to capture the leading ones and refinement bits at each bit level in each data set. Following the order given in Figure V.3, a multi-dimensional hierarchy of list structures is defined for use in the ensuing data processing stages. Within each list (as EQW progresses), the fields in the corresponding list structure are updated. The bit stream at each bit level is appended to the MUX "prepack" buffer as well as its corresponding size being added to the total in the bit packing information field. The bit level position where processing begins is put in the high bit information field. The total packed list size is placed in the total packed field. Finally if more than one internal processing scheme is used (e.g. NULL transform mode for bi-level information or other internal modes of operation), the scheme used for the list is placed in the scheme field.

Each MUX list [redacted]ully independent, implicitly contains a[redacted] set of statistical information available for determining the amount of data to pack for each list and is optimal for organizing the final compressed bit stream.

Data Priority and MUX Control

The information contained in each list structure is used to organize the final bit stream based on end user compression requirements. DAC has implemented numerous MUX modes o operation. Two common modes of operation are outlined in this Section.

Signal to Noise Ratio (SNR) Progressive Mode

This mode of MUX places priority on the data such that the final ordering optimizes the SNR given a user specified image compression size. For the discussion that follows, assume that the user has specified the desired compression ratio in some manner. Assume that an optimal bit budget distribution scheme is in place. Thus the number of bytes that has been allocated to each color channel is known (more will be said on how to determine the optimal bit budget distribution for each channel using the MUX later). The information fields in the MUX list structures are used to determine the amount of data to pack for each list such that the SNR is optimized for the specified bit budget. A psuedocode explanation of this technique appears in Figure 35.

Fields that refer to the MUX list structure appear in italics while other local variables appear in normal typeface. Basically the idea is to loop through each list checking to see whether the current processing bit level is equal to the bit level of the list under consideration. If the two bit levels are equal, the channel bit budget is decreased by the amount of data available in this particular list for the bit level in question (from MUX field *pliPackingInfo[cCurBitLevel]*. The MUX field *liCurBytesCount* is incremented by the number of bytes available. Additional processing takes care of the remaining bits. The remaining bits will be considered in the channel bit budget on the next visit to this particular list. The MUX field *cCurBitLevel* is decreased by one such that it will again be enabled when the bit level in the main loop is decreased by one.

This procedure optimizes the SNR since it processes the largest levels in each list first. Thus information for the largest coefficients throughout individual decomposition data sets is sent first according the natural processing order outlined earlier. Once the channel loop exits, each MUX list will contain a field indicating the amount of data to pack into the final bit stream in each case.

Resolution Progressive Mode

The resolution progressive mode of bit budget distribution is a simple extension of the psuedocode implementation of the SNR mode of the previous Section. In order to obtain a resolution progressive mode, the '*Waveket Transform level*' for loop is taken outside of the main *BitBudget / BitPlane* while loop such that resolution level is given priority. In this manner, a lower resolution image can be reconstructed in the inverse wavelet transform stage. The resolution of the ensuing image depends upon the number of levels required in the end user requirements.

Data Packing Using the MUX Lists

Composing the final bit stream is a simple matter once the bit budget distribution scheme has run to completion. The packing technique can follow one of the normal processing orders outlined earlier (e.g. color level priority or level priority color interleave) or another access technique as required by the end user. The amount of data to pack for each list has been determined in the bit budget distribution stage. If the data to be packed for a given list is greater than zero, then the total data size, scheme and high processing bit level are packed as header information into the final bit stream. If the data to be packed for a given list is zero (i.e. not required), then a smaller header is packed to indicate a zero length list. Currently a register type approach is being developed to handle zero length lists. All lists that have data available will set a bit position in the list register so that the decoder can determine which lists have made a contribution. Empty list will be flagged with a 0 bit in the list register. The list register is packed in the final code-stream. This technique will save the size of the list header overhead of empty lists.

Overhead Consid[redacted]on of the MUX Architecture

There is a small overhead placed the compressed data by the MUX implementation. There are three header fields to pack for each MUX list.

- The total bytes packed.
- The scheme used to process the list.
- The highest bit plane for data in list.

Note that if there is only one internal scheme used to process the entire image, then the scheme field does not need to be packed for each list.

In order to calculate the cost of using a MUX in terms of header sizes for each list, consider that a 24 bit color image of size 2048 by 2048 needs to be processed in a lossy fashion (i.e. the working MUX model of Figure 28). Assume that the size of the smallest wavelet transform level is 8 by 8. Thus there are 8 decomposition levels to consider in forming the MUX list hierarchy. Assume that the wavelet decomposition data has been converted to a 16 bit integer representation. Also assume that there is only one internal processing scheme for the lists. Given these initial conditions, a worst case analysis is conducted to determine the header size for each list. The results are tabulated in Table V.1 for the Y-Channel wavelet decomposition.

From table in Figure 36, the total Y-channel header size for this particular image is 443 bits. A similar calculation can be conducted on the U/V-Channels (assuming one less decomposition level for each) to yield 368 bits each. The original raw image size is 2048 x 2048 x 3 bytes. Thus the MUX packing overhead in this particular example is about 1 header bit for every 854 bits of compressed data. In the lossless case where each channels has the same number of decomposition levels (or correspondingly, in the 8 bit grayscale case), the overhead is still very small at 1 header bit for every 757 bits of compressed data packed.

The table in Figure 37 outlines the overhead size for square images with dimensions that are a power of 2 beginning at 16 by 16 and ending at 64k by 64k. Note that the overhead size if 44 bits in both the lossy and gray scale cases. The reason for this is that the wavelet transform is not used on the U/V channels when the original image dimensions are 16 by 16. The Y-channels is decomposed once. A plot of percentage overhead versus image dimension is given in Figure 38. Both lossy and lossless cases

appear on the sa[redacted]plot. Note that the image size can be [redacted] small while still maintaining a relatively low overhead in terms of the total MUX list header sizes.

Bit Budget control from the MUX Architecture

One of the most difficult tasks that must be addressed in any compression scheme is scalability. The internal procedures must be developed such that the image under consideration can be compressed to a user specified size in an optimal manner. The task is compounded for color images since there are three channels to consider. The major problem is how to distribute the user specified bit budget between the three channels.

Standard color transforms such as YUV or YIQ are normally used to redistribute the RGB color information prior to the multi-resolution decomposition stage of the compression procedure. These transformations concentrate most of the energy into one channel making them better suited for compression. Another color transform that is gaining popularity is the KL transform. This particular transform technique is based upon a principle component analysis (PCA) implemented on the original raw color information in order to determine the optimal color redistribution for the three RGB channels. Generally speaking, after completing a color transform stage, two of the three channels are down sampled by a factor of 4 to 1 for lossy compression. In doing this, the amount of data that must be compressed is reduced by a factor of 2 with minimal loss in visual quality for the reconstructed image.

The easiest bit budget distribution scheme to implement is one that follows the color transform down sampling ratios. If the transform under consideration is YUV-411, then 4 parts are allocated to the Y-channel for every 1 part allocated to the U / V-channels respectively. However, the energy distribution of the wavelet transform data generally does not follow this strict down sampling guideline. Instead it varies from one image to the next.

DAC has developed a dynamic bit budget control architecture that is based on the implicit information contained in each MUX list. In this Section, a bit budget allocation procedure will be introduced that can be used for optimal scalability in normal processing modes of operation. This technique is based implicitly on the amount of energy contained in each color channel of the wavelet decomposition data

sets. A data ratio concept is used together with the prepack information contained in the MUX list structures to determine the budget for each channel.

Compression procedures that process wavelet coefficients in a bit plane order are implicitly tracking the energy contained in the decomposition. The most significant bits (MSBs) of the largest coefficients are packed first followed by refinement bits and the MSBs that are significant for coefficients at the next bit level. This process is repeated in a recursive fashion until the desired compression size is obtained.

Let y_{ij} , u_{ij} and v_{ij} , be the total data available in a particular orientation level for the three color channels. Then the total amount of data available in the Y-channel decomposition Y_t is obtained by summing the individual totals. A similar procedure can be followed to obtain U_t and V_t.

$$\begin{aligned} Y_t &= y_{N4} + \sum_{i=1}^{N}\sum_{j=1}^{3} y_{ij} \\ U_t &= u_{N4} + \sum_{i=1}^{N}\sum_{j=1}^{3} u_{ij} \\ V_t &= v_{N4} + \sum_{i=1}^{N}\sum_{j=1}^{3} v_{ij} \end{aligned} \qquad \text{(Eq.V.1)}$$

The next step is to calculate the pack ratios to be used for each channel of the wavelet decomposition hierarchy. The pack ratios are determined by taking the ratio of the two largest amounts of data to the smallest amount of data. Let the three pack ratios be denoted R_y , R_u and R_v. The smallest data size will be in either the U or V channels because of the down sampling step. Note that one of the pack ratios will be unity.

$$\begin{aligned} R_y &= \frac{Y_t}{MIN(U_t,V_t)} \\ R_u &= \frac{U_t}{MIN(U_t,V_t)} \\ R_v &= \frac{V_t}{MIN(U_t,V_t)} \end{aligned} \qquad \text{(Eq.V.2)}$$

The pack ratios given in Eq.V.2 are used to determine the optimal amount of data to allocate for each color channel based on the user specified compressed file size. However, any additional overhead introduced into the final bit stream by the MUX must be taken into consideration. Let yh_{ij}, uh_{ij} and vh_{ij} be the individual header sizes in each wavelet channel for a particular orientation level. Then the total header sizes for each wavelet channel are obtained by summing the individual totals.

$$\begin{aligned} Y_h &= yh_{N4} + \sum_{i=1}^{N} \sum_{j=1}^{3} yh_{ij} \\ U_h &= uh_{N4} + \sum_{i=1}^{N} \sum_{j=1}^{3} uh_{ij} \\ V_h &= vh_{N4} + \sum_{i=1}^{N} \sum_{j=1}^{3} vh_{ij} \end{aligned} \quad \text{(Eq.V.3)}$$

If doing ROI processing, the packing overhead introduced by the mask must be taken into consideration in determining the bit budget for each channel. Let the total pack size for the ROI mask be denoted as M_t. The logical approach to take is to make an adjustment in the bit budget for each channel based upon the pack ratios of Eq.V.2. In this manner, the mask overhead is distributed proportionally to each wavelet channel. The unit adjustment factor U_a is determined from the total mask overhead and the total pack ratio.

$$U_a = \frac{M_t}{R_Y + R_U + R_V} \quad \text{(Eq.V.4)}$$

The adjustment size for each channel is determined by using the result of Eq.V.4 and the pack ratios of Eq.V.2.

$$\begin{aligned} Y_a &= R_Y \cdot U_a \\ U_a &= R_U \cdot U_a \\ V_a &= R_V \cdot U_a \end{aligned} \quad \text{(Eq.V.5)}$$

A useful set of calculations that comes out of Eq.V.1 to Eq.V.5 is the minimum compression bits per pixel (Bpp_{Min}) and the maximum compression bits per pixel (Bpp_{Max}). These values are calculated below (assume that the original raw image file size is F_t bits).

$$Bpp_{Max} = \frac{8}{F_t} \cdot (Y_t + U_t + V_t + Y_h + U_h + V_h + Y_a + U_a + V_a)$$
$$Bpp_{Min} = \frac{8}{F_t} \cdot (Y_h + U_h + V_h + Y_a + U_a + V_a) \quad \text{(Eq.V.6)}$$

These expression represent bounding compression values available for the image under consideration. If all data is packed for each of the wavelet data sets, the result is Bpp_{Max}. If just the header information is packed, then the result is Bpp_{Min}.

Suppose the user specifies a compressed file size of F_{user}. The unit bit budget U_{bb} is determined from the pack ratios and the total header sizes.

$$U_{BB} = \frac{F_t - (Y_h + U_h + V_h)}{R_Y + R_U + R_V} \quad \text{(Eq.V.7)}$$

Now the optimal bit budget for each channel Y_{bb}, U_{bb} and V_{bb} can be determined using U_{bb}, the individual pack ratios and the overhead parameters.

$$\begin{aligned} Y_{BB} &= U_{bb} \cdot R_Y + Y_h - Y_a \\ U_{BB} &= U_{bb} \cdot R_U + U_h - U_a \\ V_{BB} &= V_{bb} \cdot R_V + V_h - V_a \end{aligned} \quad \text{(Eq.V.8)}$$

The expressions given in Eq.V.8 will always yield a near optimal bit budget for each color decomposition channel. The technique is based implicitly on the energy distribution map recorded in MUX lists and the total overhead associated with the

underlying proces[redacted]Note that if processing with no ROI, ther[redacted] variable falls out of the calculation.

This concept can be extended to form pack ratios for each subband if more accuracy is required. A similar data ratio technique can be implemented to control the bit budget in that case. The necessary information is contained in the MUX list structures. However for most practical application, the cited distribution technique is sufficiently accurate. The user specified file size can be obtained within several bytes in implementing this technique. In addition, the maximum and minimum attainable file sizes are known prior to the final packing stage.

A simple routine has been developed for distributing the final bit budget in the level where it is determined it will expire. The idea is quite simple. Given that the bit budget will expire on a particular bit plane in a certain transform level, then the bit budget is redistributed such that each orientation gets a proportional amount based on the amount of data that each orientation can take. The amount of data each orientation gets is determined by using ratios between orientations on the bit level under consideration. This is an important consideration since the net effect is approximately a 1 dB improvement in the PSNR.

Using the MUX for Region Processing

So far the focus has been on the design of a MUX control architecture for normal image processing modes of operation. The main highlights are the data processing orders, the MUX list structure, the bit budget control technique, and organizing the final bit stream. DAC has extended the MUX concept to include a variety region processing modes of operation.

Region Processing Orders

The processing orders introduced in the previous Section are specific to the normal processing modes where one or more ROI are not specified. DAC has implemented a region processing design based on the natural extension of the MUX concepts introduced in the previous Sections. Both lossy and lossless processing orders are again considered.

General Color Region Level Processing Order

The region level processing order places priority on each ROI in a descending fashion followed by the normal level priority scheme of the original MUX design. Inherent in the MUX scheme is the general assumption that data of similar magnitude in each succeeding region is less important to the overall image reconstruction. A secondary priority key is placed on the wavelet transform level. Data of similar magnitude contained in a lower resolution level is more important to the reconstruction procedure than data at a higher resolution level. The general processing order for each color channel (assuming a 4 region priority scheme) is given in Figure 39 where R_i is the region level for i = 1,2,3,4.

Region Level Color Processing Order (Lossy)

The individual orders of Figure V.10 can be combined into a single processing order. The new order is obtained by interleaving the expressions making use of the inverse color and wavelet transformation relationship that is presented for the normal MUX modes of operation above. The result is illustrated in Figure 40.

The intrinsic 4-1-1 color down sampling relationship is maintained for region processing MUX modes. In terms of the reconstruction process, the complete level 4 Y-channel information (i.e. LL_4-HL_4-LH_4-HH_4) is required together with the U and V channel low pass information (i.e. LL_3 in each case) for each region under consideration.

Color Interleave Region Level Processing Order (Lossy)

The color interleave processing order extends naturally to the MUX region processing modes. The order is illustrated in Figure 41. This type of ordering may be more suitable for certain applications that require the data to be encoded for a progressive download based on regions of importance. As in the previous case, the down sampling relationships that exist for the inverse color and the wavelet space are maintained.

Region Color Processing Orders (Lossless)

As in the lossy color region processing MUX modes, similar processing orders can be outlined for the lossless case where full data sets exist for all color channels in each

region. In this case corresponding decomposition level orientation information in each channel is related in the original color transform space for each region under consideration. These data are processed and grouped together in the wavelet transform space and are ultimately put in the final bit stream by the MUX control architecture.

The processing order for lossless color region processing is illustrated in Figure 42. Note that in each case, the full decomposition data set for each channel is maintained. The inherent relationships that exist in the inverse color and wavelet space are maintained for each region channel.

DAC has developed both SNR progressive and resolution progressive MUX modes of operation for region processing. In addition, several specialized MUX modes were developed that may be useful in certain applications. The regions can be defined automatically by the technique described in Chapter III, or user defined ROI can be used to group the data. User defined ROI can have arbitrary shape or can be defined in terms of simple geometric elliptical or rectangular region primitives.

One of the specialized MUX region processing modes can be introduced at this point since it fits into the context of the ordering discussion.

Transparent Region Color Processing Orders (Lossy)

This particular region color processing order is useful for applications that require transparent region channels that has little or no influence on the ordering of the data. Either of the normal SNR and resolution progressive modes are overlaid with a complete region channel description. The technique is implemented simply by taking the region index to the inner processing loop in the pseudo code implementation example of Figure V.8. This ordering technique may be useful in video processing applications where a complete region mask description is required together with a compressed frame of information. The region description may be used to process subsequent frames in the sequence. One of the processing orders that falls into this MUX mode of operation is illustrated in Figure 43. In this case, the high bit plane of each region list is processed first causing the region classification of all coefficients to be transparent to the distribution and packing routines.

Transparent Regio[redacted]lor Processing Orders (Lossless)

Corresponding transparent region processing modes exist for the lossless case. A typical example is given in Figure 44. As in the lossy case, the final bit stream is organized independent of the constraints of the region channels which may be useful in certain cases.

MUX List Structure for Region Processing

The MUX organizational list structure concepts introduced for normal processing modes have been extended to include many region processing modes of operation. In this case, the total number of lists is a function of the number of ROI. If there are 4 ROI, then there are 4 times as many MUX lists. However, the basic operation of the MUX control architecture is similar in each case.

There are three high level modes used to categorize the MUX architecture.

- Normal processing mode (ROI disabled).
- Region processing mode (ROI only).
- Mixed processing mode (normal and ROI enabled).

The next Section outlines the operation of the bit budget and MUX controls for many of the region processing modes developed at DAC. The mixed processing modes are briefly discussed later in the Chapter.

Bit Budget Control for Region Processing MUX Modes

Transparent Region Level Color Mode

Transparent region processing is mentioned above. In that particular ordering example, the region index is placed in the inner most loop in the bit budget distribution function. The processing index placement is basically a method of incorporating a transparent region layer of processing into the normal MUX modes outlined earlier in the Chapter. Processing and packing in this mode gives ROI a low priority.

Distributing in this context ensures that the SNR progressive and resolution progressive MUX modes operate as they did before. The exception in this case is that there is a variable length region mask overhead that must be taken into consideration.

The bit budget distribution functions are calculated as they were the normal MUX processing modes. Based on the file size (or resolution level) requirements, each color channel receives a proportional amount of the bit budget based on the ratio distribution technique used in the MUX architecture.

The DCT auto regions and arbitrary or primitives user defined region types can be operated in this mode. In DAC's internal processing architecture the user must select the region coverage technique used to categorize the wavelet coefficients. DAC has implemented the wavelet mask down sampling technique of the JPEG-2000 VM, and it can be used in any of the region processing MUX modes. In this manner a common mask can be used in each orientation level or the VM mask down sampling technique can be used for individual orientation level mask coverage, for both automatic or user defined ROI in lossy and lossless MUX modes of operation. The number of region categories can be selected as 2, 3, or 4 channels.

In order to test the transparent mode of operation the YUV-411 color transform is applied to a 24 bit 256 x 256 Glacier park mountain scenery image. The 9-7 kernel implemented in a lifting scheme is used as the wavelet transform. The DCT region formation technique of Chapter III is employed to generate the common masks for each wavelet transform level. The mean square error (MSE) is measured for a number of compression ratios. The result is illustrated in Figure 45 in a plot of MSE versus Bpp. The plot shows the break points in MSE for each automatic ROI used in the example. The rate of image degradation (i.e. MSE and PSNR) must be controlled precisely for each ROI.

The MUX overhead is calculated based on the mask type and process selections. For arbitrary user constructed masks, the mask file can be loaded to guide the MUX. In the 4 ROI common mask case, the overhead is 0.5 Bpp in packing the entire mask. In the VM mask case, the overhead is 2.0 Bpp. If simple primitives are used to form the user mask instead of an arbitrary shape, the overhead is greatly reduced. The arbitrary mask overhead can be reduced with the addition of an entropy coding stage. The common DCT mask overhead is cited in Chapter III as approximately 1Kb (0.2 Bpp) for an 8 bit 256 x 256 image size. These are rough estimates that do not take the

header sizes into [redacted]unt. However, they do serve as a compa[redacted]e guideline for the current discussion.

Region Priority Level Color Mode

In this mode of MUX operation, the pack ratios for each wavelet channel are determined as in the normal processing mode. However, there is one important difference. In this particular mode of operation pack ratios are determined for each region channel. This implies that there are 12 pack ratios for a 4 ROI process. The bit budget distribution is based on the overall wavelet channel pack ratios and the ROI data totals in each region channel. The distribution function also takes the mask and list header size information into account in the overall calculation.

This mode of MUX operation is designed to distribute the MSE and the PSNR image reconstruction measurements in an approximately uniform manner for all region channels. A proportional amount of overall bit budget is allocated to each region channel based on the region and color channel pack ratios. Figure 46 illustrates the result of using this mode of operation for the Glacier park image. Notice how the quality of each region channel degrades in comparison to the others. Auto detected DCT common masks are used to generate the result.

Absolute Region Priority Level Color Mode

In this mode of operation, absolute priority is given to the region channels. The distribution function is the same in this case in that the bit budget is divided into 12 according to total data ratio technique mentioned earlier. However the bit budget is distributed in a biased fashion giving the highest priority to the most important ROI. Some regions may not receive a budget based on the compression requirements. The bit budget is distributed region by region beginning at the most important ROI. Thus complete sections of the original image can be eliminated altogether if the bit budget is small. If only the important regions of an image need to be saved, this technique can be employed to partition the image.

A plot of MSE versus Bpp for the absolute region priority color MUX mode is given in Figure 47 for the Glacier park image. Notice how the regions fade out very fast and sharply. This occurs when the region no longer has any budget allocated to it. At

that point, the re[redacted] is basically invalidated in terms of a[redacted]ontribution to the reconstructed image. That portion of the image basically fades out. Note that the amount of quality of the fade out depends on the down sampling technique to translate the masks. The VM mask formation technique causes a very gradual fade out to occur. In the common mask approach, the effect is much more abrupt.

Scaled Region Priority Level Color Mode

In this mode of MUX operation degradation rate of each ROI can be controlled. Each ROI contributes to the final bit stream. However, instead of having each ROI degrade at a uniform rate (as Section V.5.3.2.), the quality measurements of each succeeding can be controlled. The initial bit budget for each region channel is calculated as before in that the specified compressed file size is split into 12 bit budgets to be spread between each of the 4 region channels. However after the initial split, the allocation amounts are changed heuristically based on a priority factor that is set for each ROI. For example, suppose it is decided that ROI 1 is 50% more important than ROI 2, ROI 2 is 30% more important than ROI 3 and ROI 3 is 20% more important than ROI 4 (the background). Using this assumption as a starting point, the amount of data allocated to each region channel is changed slightly. The net effect is to cause the quality measurements in each region channel to degrade at slightly different rates. A plot illustrating this effect is given in Figure 48. Note how the MSE break points for regions 2 and 3 have shifted slightly towards the left.

The result of using this particular MUX mode illustrates an important property that should be available for any ROI processing technique. The problem with many ROI processing techniques is that it is difficult to control how much each region contributes to the final bit stream. The technique outlined here can be used not only to implement this effect, but also to control the degradation for each region channel based solely on an importance factor associated with each ROI.

Region Percentage Priority Level Color Mode

In this mode of operation the user can specify a data percentage for each region based on the total amount available in each region channel. The distribution function is slightly different in this case. The data totals are determined for each region channel along with the wavelet color channel totals. There are still 3 bit budget for each

region channel. In this case however, the amount of data to include for each region channel can be set by the user as a percentage of the total in each case.

Currently when this mode of operation is invoked in DAC's compression engine, the user can cycle through the operation until the desired size for each region is obtained. The data totals are displayed after each run. Note also that in using this mode of operation, one or more ROI can be eliminated or faded heavily as in the absolute region priority MUX mode of Section V.5.3.3.

A plot illustrating the result of using this mode of MUX operation is given in Figure 49. Initially all available data is included for each region followed by equal decrements for each region channel.

Using the MUX for Mixed Processing

So far both normal (non-region) and ROI processing modes have been discussed. In addition to these modes of MUX operation, DAC has developed mixed mode processing capabilities. The modes of operation discussed for both normal and ROI processing are extended such that they can be run simultaneously for an image under consideration. A wavelet transform level partition is conducted based on the desired number of region and the number of non-region levels.

Currently non-region levels can be defined for processing lower resolution levels and region levels can be defined for process higher resolution levels. However, the implementation is not restricted by this distinction. It can be changed to regions over non-regions or to an interlaced combination of the two. The parameter that controls this distinction is termed the region start level. It can be set to any valid wavelet transform level (it is set internally to –1 to disable region processing.) Thus the MUX technology can be used in exclusive non-region mode, exclusive region mode or a combination mixed mode of operation.

The ordering techniques outlined for normal and region processing MUX modes will not be duplicated in this Section. The same concepts apply for mixed mode processing. The bit budget distribution functions work as they did before, but in this case they exist simultaneously. In some modes of operation there may be as many as 15 bit budget definitions used to organize the final bit stream. The method used to

determine them [redacted]ot changes. The same ratio technique t[redacted]as its basis in the MUX list structure is used to determine the appropriate allocation in each case.

There is one additional feature that can be exploited for mixed mode processing. An importance factor can be attached to the non-region levels. The net effect of this parameter is to taper the amount of data included for the non-region levels. In the current implementation, the non-region levels are processed first. Thus they considered first by the distribution function. The importance factor allows the user to decrease the amount of data included for the non-region levels by a certain percentage. The delta amount is considered in the bit budget distribution for the region levels.

The region processing overhead is slightly smaller in this case. Depending on the region start level parameter, there will be less region header information required in the bit stream since there are fewer region levels. However, the header overhead for the lower resolution levels is quite small anyway.

One example is given here to illustrate the operation of mixed mode processing (see Figure 50). In this particular case SNR progressive mode is used for the upper 3 levels and scaled region priority on the bottom 3 levels. Thus the bottom level degradation rates have been adjusted to favor the most important region and attenuating in some fashion between the other 3 regions. Notice how the MSE is lower in this case with the same region overhead as the previous case of Figure 48.

DCT Region Formation as a Classification Scheme

In observing this result, it is apparent that for the same region overhead, a better result is obtained in mixed mode. There are a number of reasons for this. The first is that the region channel coverage is not an exact overlay. All masks used to group or categorize data must deal with the down sampling issue at different resolution levels, and the tight overhead restraints of the compression channel. As it appears in the results presented here the DCT region detection mode can be applied to threshold wavelet coefficients to form region channels. The amount of data packed for each channel can be controlled by the MUX. In addition the initial data split used to create the partition can be controlled before the DCT mask procedure begins.

There is some re[redacted] migration or intermingling of the coeffi[redacted]s that occurs at the boundaries where sharp changes occur in the original image. Some work was conducted in developing heuristic techniques to decrease the miscoverage in hot areas. And there is a benefit there. Furthermore the DCT low pass filtering stage affects the original priority bias used to partition the data. There will be a benefit in determining the optimal data split. By adjusting the original region partition (e.g. equally spaced regions), the plot of MSE versus Bpp can be set to partition the data in other ways. Inter-resolution coverage is another problem to consider. The mask generation technique of the JPEG-2000 VM addresses the miscoverage problem. DAC has completed some initial investigations of the VM down sampling technique and further testing is required.

There is a trade off between accuracy of each region channel and mask overhead required for region channel operation and generation of the accuracy in the first place. The DCT approach shows much promise especially on the highest resolution level where the masks are the most accurate. One of the benefits of using this the DCT approach is that it can be translated to other transform levels in the frequency domain. Or alternatively, the new masks could be generated at a different wavelet level.

Other Processing Modes

There are many other modes of region processing operation that have not been presented in this document. The whole user defined region generation and MUX modes were not included. The processing modes introduced in this Chapter can be used in the same way to control each region channel. Simple primitives are cheap in terms of overhead. One mode of operation supports full mask sets so that arbitrary mask can be loaded and sent with the data to form the region channels. The mask coverage technique can be selected as common masks (a smaller raw size that is up sampled at the decoder side) or VM masks, which retain the original image dimension. More time is required to study the effects of both techniques as well as other region formation schemes.

Experimental results for the resolution progressive modes of operation are not presented in this document. These modes of operation are not currently available. However their implementation is not difficult. These MUX modes are currently under development.

The experiment[redacted]ults presented in this Chapter were obta[redacted]using DACs 1D bit level sorting implementation. DAC is currently incorporating the 2D version based on EQW into the region processing channel. There may be benefits in retaining both techniques. For example, in bi-level image processing. Currently the EQW sorting is implemented for normal (non-region) processing MUX modes. Both sorting modes are available for lossless compression. There are numerous wavelet kernels and color transforms for both lossy and lossless compression. In addition, the number of region channels can be set with a current maximum of 4. Primitives can be used as desired. The number of primitives is not restricted to 4 with region overlaps given to the most important region.

Bit Stream Syntax

DAC's current bit stream structure is rather dynamic given the different types of organizational strategies that exist in the underlying core technology. Normal, region, and mixed processing modes in addition to arbitrary wavelet, color, entropy coding various sorting stages in both lossy and losses cases.

Currently the core architecture can be divided into 2 categories.

- Lossless compression

  Full data sets lossless color selection, lossless integer lifting scheme wavelet types, lossless sorting and packing stages, with additional entropy coding can be selected for completely lossless and slightly lossy (some of the color transforms are slightly lossy) image compression.

- Lossy compression

  In the lossy case, color selections, more wavelet kernels / lifting schemes, lossy sorting and packing stages, with variable length coding can be selected

In both lossy and lossless cases region and mixed modes can be used. Currently normal operational modes can be realized in an almost transparent fashion for many current compression schemes as well as our own internal EQW / 1D sorting. The general form for the lossy and lossless header structures is given in the tables in Figures 52 and 53.

The current architecture is very flexible for introducing new technologies. Many of the modules are completely interchangeable. The exact bit stream syntax depends largely on the mode of operation selected by the user (i.e. regions, no regions, sorting, lossy/lossless etc.) Generally normal processing modes can be selected for all transform levels, or for any number of lower resolution transform levels. Region levels can be defined for all transform levels, or the higher resolution levels used in combination with the lower resolution normal levels. At some future date, the region and the normal levels can be mixed.

Each unit list organized by the MUX has a header tag. This header tag carries the list size and the high bit plane processing level for the list. The current implementation uses 5 bytes per list. However bit packing is currently under implementation for tag headers. This will reduce this by a significant amount. Only 2 packing schemes are used at the global level so that no additional tag header information is currently required. A diagram illustrating the structure of the tag header is given in Figure 53. As other core technologies are added to the core engine or the core technology advances, other fields may be required in the tag headers.

The basic structure for normal modes of operation is given in Figure 54. The diagram illustrates a lossy pack arrangement for a color image (YUV down sampled color transform assumed) according to the packing/processing order given in Figure 30 above. In this case the code stream consists of the file header, followed by the header tags/data.

The basic structure for region processing modes of operation is given in Figure 55. The diagram illustrates a lossy pack arrangement for a color image (YUV down sampled color transform assumed) according to the packing/processing order given in Figure 40 above. In this case the code stream consists of the file header, region header, regions description and finally the header tags/data.

The basic structure for mixed processing modes of operation is given in Figure 56. In this case the first item in the code stream is the file header followed by the normal processing mode header tags/data. Next in line is the region header, region description followed by the header tags/data. Notice that there is a region start level parameter 'r' in this case. Since there is one less transform level for U/V channels in

the lossy case, th[redacted] a processing shift in the U/V channels. [redacted]s makes reference to the "level split" notion discussed in Chapter V whereby the natural 4:1:1 relationship that exists for inverse color/wavelet transforms can be maintained for internal processing and the final code stream.

Transcodability

In order to illustrate the generality and openness of RICS, two examples are shown to depict how the 'transcode' with bi-level images (JBIG) and JPEG can be handled in RICS.

JPEG transcode

Figure 57 shows how a DCT-based code can be produced in RICS. With this transcodability, it will be straightforward to write a small conversion program to transcode an old JPEG file into a JPEG 2000 file. Although it is possible to transcode a JPEG 2000 file into a JPEG file (losing some scalability), it will probably be of very little use.

VJBIG transcode

Figure 58 shows the execution path illustrating how a bi-level image such as a text document can be efficiently handled in the RICS system. The NULL transform is applied (nothing has to be done). Generally binary text documents contain mostly high frequency energy. Multi-resolution decompositions will not necessarily be a suitable basis for efficient coding. In fact, current well-known techniques specialized for binary images are applied directly in spatial domain. While using a RICS system for processing a compound document with mixed grayscale/color/text information, a set of rectangle shapes suffice to enclose the text regions. This is roughly equivalent to run-length coding in existing binary image compression techniques to skip strings of zeros. The pixels within each rectangular region are then coded into a one-dimensional stream via 1-D algorithms or JBIG routines, as discussed above.

Post Processing

For reconstructed images with wavelet based coding methods at low bit rate, de-ringing is usually a useful post processing procedure to improve (mainly) the visual

quality. Nonadaptive procedures, which apply the de-ringing filtering to all pixels without discrimination, have the following problems.

- While they can successfully remove ringing artifacts, they may also wash out details of image.
- They usually rely on too many parameters that are to be determined by human users.
- The process is time consuming.

The RICS system employs an adaptive de-ringing algorithm for post processing. Since the ringing artifacts usually appear around edges where sharp changes occur, this adaptive filtering process is applied only to edge areas. As the result, the post processing removes artifacts around edges and prevents fine details from being smoothed out by the filter. Compared with non-adaptive methods, the processing time is remarkably reduced since the filtering is applied selectively to pixels. The diagram of Figure 59 shows the algorithm of the adaptive post processing filter. First of all, the pixels of reconstructed image are classified into edge pixels and non-edge pixels. Edge pixels are enlarged into edge regions since the ringing artifacts do not occur right at the edges but around the edges. Then the artifact removal filter is applied only to the edge areas.

We did some tests on this adaptive de-ringing procedure. For the filtering stage we used the filter described in the JPEG 2000 VM 3.0 (B). Figure 60 shows the result of the modified post processing. Artifacts are clearly visible in the first picture especially behind the cameraman's back and around the tripods. As we can see from the second picture, artifacts are removed, however, the details in the grass field and on the man's pants are also removed. In the last picture, the modified filter successfully removes artifacts and at the same time retains the details. Figure 61 shows the edge area (in white color) that was used in producing the third picture.

As the result of using this adaptive filtering, processing time is greatly reduced since less pixel are processed. The table in Figure 62 shows the performance comparison of the VM 3.0 (B) post filtering and the RICS adaptive filtering

Edge Detection

To determine if [redacted]el is an edge pixel, the average power [redacted]rence between the pixel and its neighboring pixels are measured against a threshold (bottom threshold). If the average power difference of a pixel is above this threshold, the pixel is marked as an edge pixel. The threshold acts as a high pass filter that filters out the pixels with low power levels (Band pass filters can also be used for edge detection by introducing a proper ceiling threshold).

Edges are thickened to form edge areas after the edge pixels are detected. With the right choice of the threshold, this modified filtering process can be applied to reconstructed images with compression ratios as low as 8:1. Since for most wavelet coded images, no artifacts can be visually detected for compression ratio lower than 8:1, our results suggest that this filter can be applied to most reconstructed pictures regardless of its compression ratio.

Parameters Optimization

One of the problems with Shen's filter is that there are many parameters associated with the filter, which makes the filter difficult to use. Reducing the number of parameters will improve the usability of the post processing filter.

There are three different potential functions implemented in VM 3.0 (B) for controlling the post processing: Quadratic Truncation, Huber, and Lorenzian. Our test results shown that the potential functions Huber and Lorenzian do not outperform the Quadratic Truncation in any of the 62 cases for PSNR measurements. Furthermore, visually Lorenzian creates many noticeable ghost images and shadows, and Huber does remove artifacts successfully however it blurs the image more than Quadratic Truncation.

Computation Speed

For evaluating the three potential functions, the following computation needs to be calculate $n \times (n-1)/2$ times for every pixel (where $n = \textit{filter size} \times 2 - 1$).

Lorenzian:	1 float division, 2 float multiplication, 1 float addition and 1 float
	log operation

Huber: [redacted] 1 comparison, 1 float multiplication o[redacted]loat multiplication with 1

addition

Quadratic Truncation: 1 comparison and 1 float multiplication

In our tests, Quadratic Truncation is the fastest and it performs the best for artifact removal with the least degradation to image quality. Therefore, the other two potential functions will not be used.

Threshold (γ parameter in the potential function)

The γ parameter is examined using the quadratic truncation potential function (default value is 16), given by:

$$\hat{e} = \arg\min_{x_j} \sum_{x_i \in N} \rho(X_i - X_j)$$

$$\text{where, } \rho = \begin{cases} \gamma^2 & ,(x_i - x_j) > \gamma \\ (X_i - X_j)^2 & ,(x_i - x_j) \le \gamma \end{cases}$$

We can see from the equations that only pixels with similar neighboring pixels are affected when varying γ. When γ is decreased, there will be more γ^2 as the result of ρ and the intensity of similar pixels output image will be more uniformed resulting a more blocky looking type of image (less gradual) for regions with similar intensities. Experiments were performed with 2 images (one grayscale and one color image). The table in Figure 63 shows the result for the color image.

We can see from the results that the higher the γ, the worst the image quality. However, the difference in image quality is not very significant. The pictures look very similar and there are no significant changes that can be detected. However as predicted, the blocky looking clouds (see Figure 60) are seen when $\gamma = 8$ or below. Therefore by setting γ lower we can increase the image quality by some small amounts without changing the visual quality of image. Therefore $\gamma = 12$ was chosen to be the default value for γ parameter in the potential function.

Filter Length

Filter length (F) determines of the size of samples collected for pixel estimations. It is obvious that the larger the filter length the longer the processing time. The default length given by VM3.0 (B) is 9 pixels. Experiments are performed with varying filter lengths to examine how the length affects the image quality (MSE and PSNR) and the artifact removal ability. Two pictures are used in this experiment (one color and one grayscale). The results are shown in the following tables in Figure 64 and 65.

As we can see from the experimental results, the small the length, the better the image quality and the shorter the processing time. However, artifacts are not completely removed for length = 5 (or smaller) at high compression ratios. The default value for the filter length is chose to be 7 pixels to increase image quality and to decrease processing time without degrading the ability for artifact removal.

Constraint

The value of constraint (Th1) affects the filter as described in the equations below.

$$y = x + c(d, Th1)$$

$$d = \hat{x} - x$$

$$c(d, Th1) = \text{sign}(d) * \text{Max}(0, \text{abs}(d) - \text{Max}(0, 2*(\text{abs}(d) - Th1)))$$

From the equations above, we can see that as Th1 increases, more pixels will have 'd' as the output for $c(d, Th1)$. Therefore, as Th1 increases, more estimate value of x will be used as the final estimate and the image quality will decrease while the smoothness of an image will increase. In order to keep the image quality as high as possible, Th1 should be kept as low as possible. However, artifacts might not be properly removed if Th1 is set too low. Three images were tested with different level of constraints in the experiment. The test results are given in the tables in Figures 66 to 68.

These results show that image quality decreases as constraint increases. However, some artifacts (very small, can only be seen when the picture is enlarged) are not completely removed when C is equal to 4 or less. In order to achieve the best image quality while successfully remove artifacts, the value 8 is suggested to be the default value.

Iteration

It is obvious that processing time is directly proportional to the number of iterations. However, artifacts might not be removed successfully if the image is not processed enough times. Figure 69 clearly shows that image quality decreases as the number of iteration increases. However, when image is enlarged, small artifacts can be still visible for R1. In order to achieve the best image quality while successfully remove artifacts, 2 iterations is suggested to be the default setting.

Mask Shape

Different shapes of masks are discussed in Shen's paper, however, only the one with the shape of a + sign was implemented in VM3.0 (B). Different masks were experimented and the shape of mask is best left unchanged since simplifying the mask degrades the performance of the filter and complicating the mask increases processing time greatly.

Using the Modified Post Processing Filter

The adaptive de-ringing is implemented based on the post processing filter in VM3.0 (B). Few parameters have been eliminated and default values have been established to increase the usability of the post processing filter.

To use the modified post processing filter:

Usage: post2 -s width height bpp -i infile -o outfile [-l mask lower threshold] [-w mask width] [-t thresh] [-f f_length] [-c constraint] [-r iterations]

Default values will be used for the parameters if the parameters are not specified in the command line. The default parameters are:

mask lower threshold: lower threshold for edge detection [30]
mask width: width of mask [12]
thresh: γ parameter in the potential function (for estimation) [12]
f_length: filter length [7]
constraints: constraints using in the clipping function [8]
iterations: number of iterations [2]

The modified post processing filter performs well with the above default parameters, however, these parameters can be changed at the command line if the user wishes. The post processing filter can now be applied to pictures with 8:1 compression ratio

with very small [redacted]ease in MSE. The modified post pro[redacted]ng filter seems to improve image quality for compression ratio beyond 11:1.

In practice, there are two ways for the user to play with the settings: on the encoder side and on the decoder side.

Decoder end

The decoder will have the full control of the post processing and the of the associate parameters. The encoder will have no control on how the images will look at the decoder end. Accordingly, there will be no addition to file header since no post processing parameters will be included.

Encoder end

When necessary, the encoder can also predetermine the post processing filter parameters. Post processing filter parameters will be stored in the image header, and the image will be restored according to these parameters. In this setting, the user at the encoding end knows exactly how the image will look at the decoder end. However, adding these parameters in the header will increase the size of the compressed image.

Total of 6 parameters will need to be packed in the header: mask lower threshold, mask width, estimation threshold, filter length, constraints and number of iterations. The table in Figure 70suggests range limits on the parameters to reduce the header size.

In total, three bytes will be needed to include these parameters in the header.

WE CLAIM

1. A region-based method for encoding and decoding digital still images to produce a scalable, content accessible compressed bit stream comprising the steps:

decomposing and ordering the raw image data into a hierarchy of multi-resolution sub-images;

determining regions of interest;

defining a region mask to identify regions of interest;

encoding region masks for regions of interest

determining region masks for subsequent levels of resolution; and

scanning and progressively sorting the region data on the basis of the magnitude of the multi-resolution coefficients.

2. An apparatus for the region-based encoding and decoding of digital still images that produces a scalable, content accessible compressed bit stream comprising:

a means of decomposing and ordering the raw image data into a hierarchy of multi-resolution sub-images;

means of determining regions of interest;

means of defining a region mask to identify regions of interest;

means of encoding region masks for regions of interest;

means of determining region masks for subsequent levels of resolution; and

a means for scanning and progressively sorting the region data on the basis of the magnitude of the multi-resolution coefficients.

3. A region-based system for encoding and decoding digital still images that produces a scalable, content accessible compressed bit stream and comprises the steps:

decomposing and ordering the raw image data into a hierarchy of multi-resolution sub-images;

determining regions of interest;

defining a region mask to identify regions of interest;

encoding region masks for regions of interest

determining region masks for subsequent levels of resolution; and

scanning and progressively sorting the region data on the basis of the magnitude of the multi-resolution coefficients.

4. A method for encoding and decoding digital still images to produce a scalable, content accessible compressed bit stream comprising the steps:

decomposing and ordering the raw image data into a hierarchy of multi-resolution sub-images;

setting an initial threshold of significance and creating a significance index;

determining an initial list of insignificant blocks;

forming the list of significant coefficients by encoding a significant map using a quadtree representation;

recursively reducing the threshold values and repeating the encoding process for each threshold value; and

transmitting refinement bits of significant coefficients.

5. An apparatus for encoding and decoding of digital still images that produces a scalable, content accessible compressed bit stream comprising:

a means of decomposing and ordering the raw image data into a hierarchy of multi-resolution sub-images;

means for setting an initial threshold of significance and creating a significance index;

means for determining an initial list of insignificant blocks;

means of forming the list of significant coefficients by encoding a significant map using a quadtree representation;

a means of recursively reducing the threshold values and repeating the encoding process; and

transmitting refinement bits of significant coefficients.

6. A method of decoding digital still images to produce a scalable, content accessible compressed bit stream comprising the steps:

decoding the bitstream header;

determining the initial threshold values and the array of initial significant pixels, insignificant bits and wavelet coefficients;

decoding the significance maps;

modifying the significance lists and decoding the refinement bits for each threshold level;

reconstruct the wavelet coefficient array;

perform the inverse wavelet transform; and

reconstruct the image.

7. A method of transmission of digital signals that creates a scalable, content accessible bitstream comprising the steps:

pack the most significant bits of the largest coefficients first followed by refinement bits and the most significant bits that are significant for coefficients at the next bit level;

repeat this process in a recursive fashion until the desired compression size is obtained;

calculate the pack ratios to be used for each channel of the wavelet decomposition hierarchy by taking the ratio of the two largest amounts of data to the smallest amount of data;

determine the optimal amount of data to allocate for each color channel based on the user specified compressed file size; and

if performing region of interest processing, consider the packing overhead introduced by the mask when determining the bit budget for each channel.

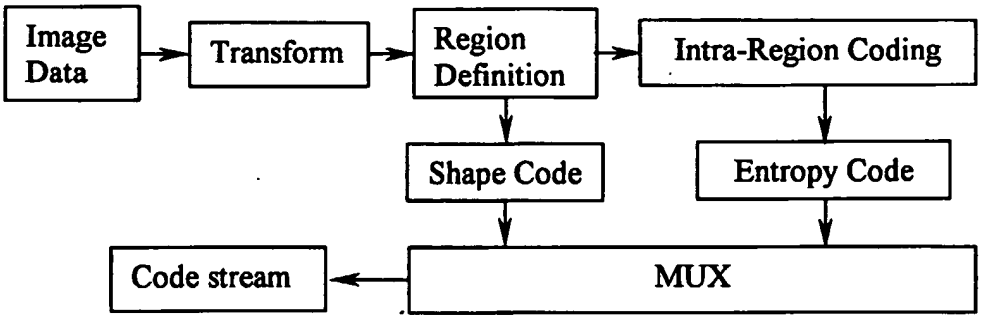


Image Data
Transform
Region Definition
Intra-Region Coding
Shape Code
Entropy Code
Code stream
MUX

FIGURE 1

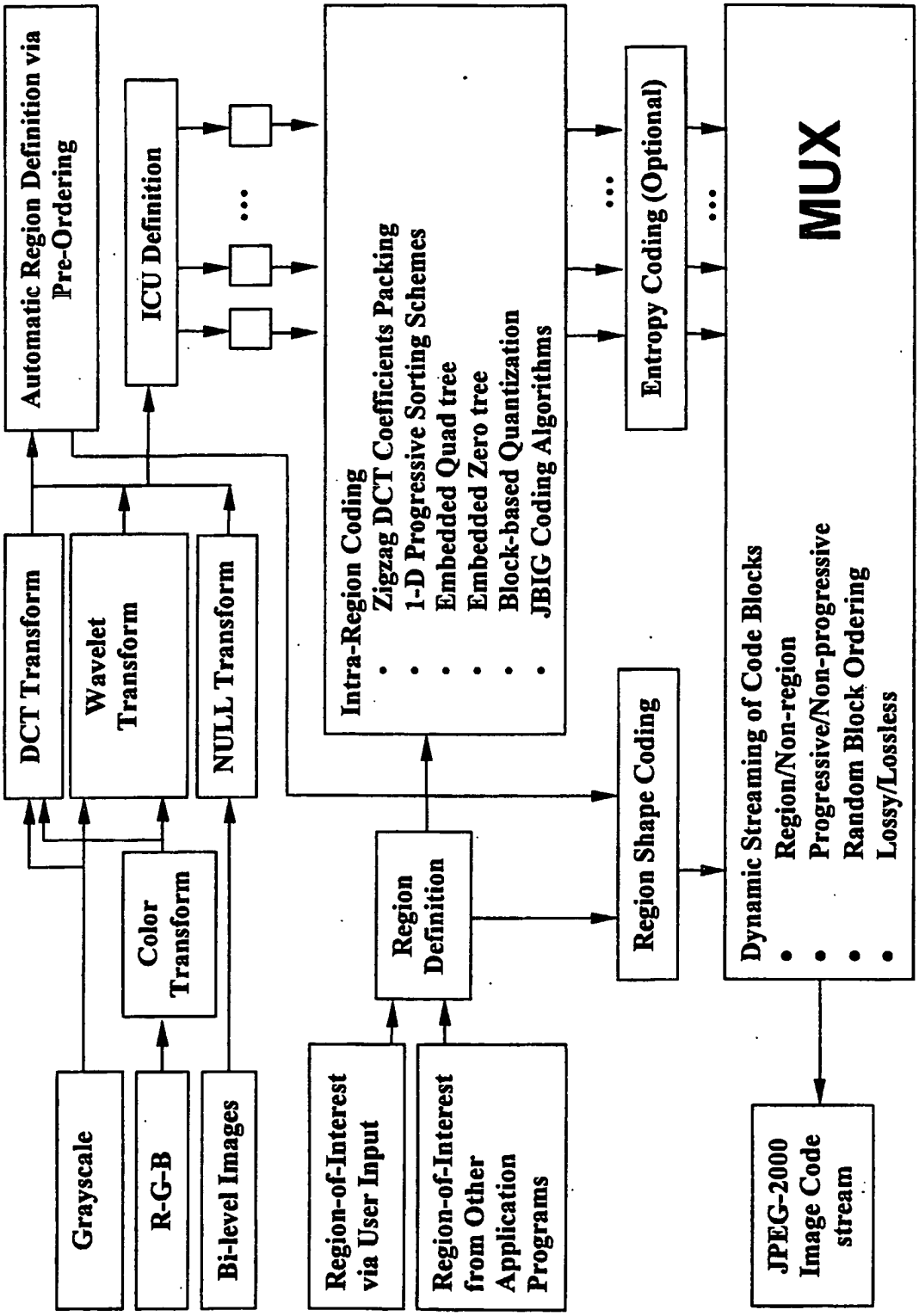

Automatic Region Definition via Pre-Ordering
ICU Definition
DCT Transform
Wavelet Transform
NULL Transform
Color Transform
Grayscale
R-G-B
Bi-level Images
Intra-Region Coding
• Zigzag DCT Coefficients Packing
• 1-D Progressive Sorting Schemes
• Embedded Quad tree
• Embedded Zero tree
• Block-based Quantization
• JBIG Coding Algorithms
Entropy Coding (Optional)
Region Shape Coding
Region Definition
Region-of-Interest via User Input
Region-of-Interest from Other Application Programs
MUX
Dynamic Streaming of Code Blocks
• Region/Non-region
• Progressive/Non-progressive
• Random Block Ordering
• Lossy/Lossless
JPEG-2000 Image Code stream

FIGURE 2

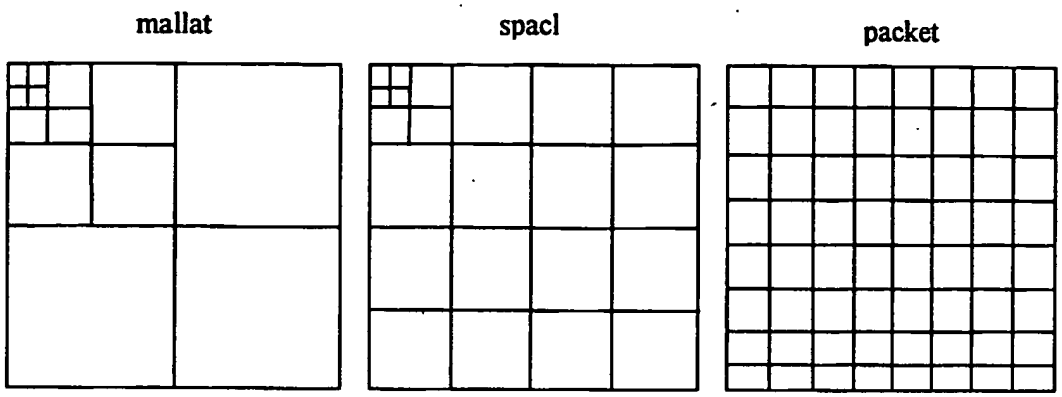


FIGURE 3

Wavelet Family	Wavelet Type (Filter Length)							
Haar	Haar (2)							
Daubechies	Db4 (4)	Db6 (6)	Db8 (8)	Db10 (10)	Db12 (12)	Db14 (14)	Db16 (16)	
Coiflet	Coif1 (6)	Coif2 (12)	Coif3 (18)	Coif4 (24)				
Symmlet	Sym2 (4)	Sym3 (6)	Sym4 (8)	Sym5 (10)	Sym6 (12)	Sym7 (14)	Sym8 (16)	
Biorthogonal	Bior1.1 (2)	Bior1.3 (6)	Bior1.5 (10)	Bior2.2 (6)	Bior3.1 (4)	Bior3.3 (8)	Bior4.4 (10)	Bior5.5 (12)
Biorthogonal (Villasenor)	Bior9 (10)9/7	Bior10 (14)	Bior11 (10)	Bior12 (6)	Bior13 (6)	Bior14 (10)	Bior15 10/18	

FIGURE 4

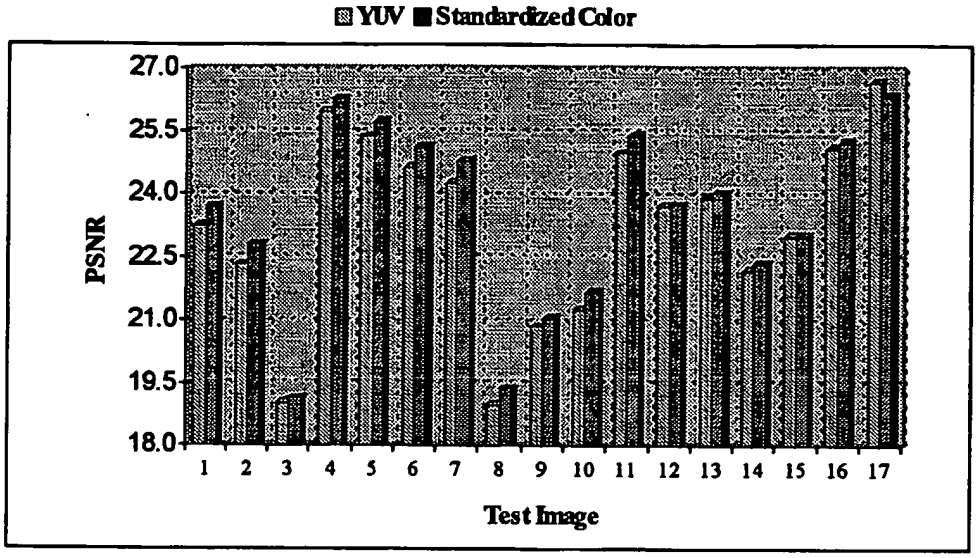


FIGURE 5

Shape	Coding Parameters	Availability in RICS
Rectangle/Square	(x_{min}, y_{min}) and *(width, height) etc.*	present
Circle/Ellipse	(x_0, y_0) and r etc.	present
Polygon	$n, (x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$	Under development
Cubic parametric curves	$n, (x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$	Under development

FIGURE 6

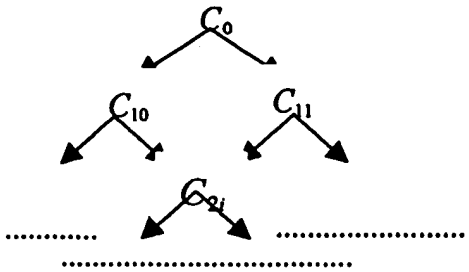
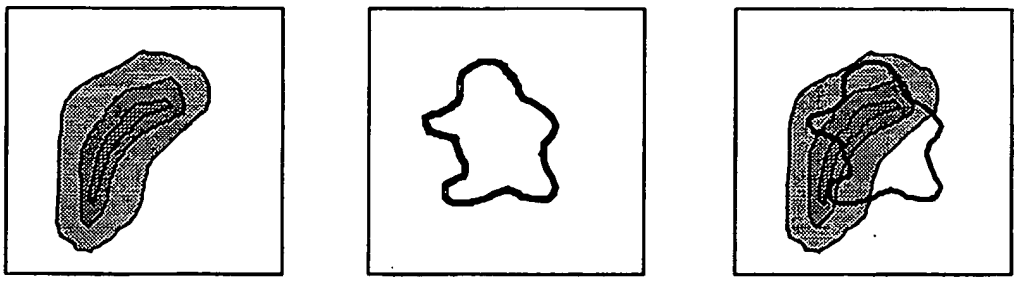


FIGURE 7



Region Mask | Dynamic Object | Extracted Dynamic Regions

FIGURE 8

HL_1 LH_1 HH_1

FIGURE 9

FIGURE 10

FIGURE 11

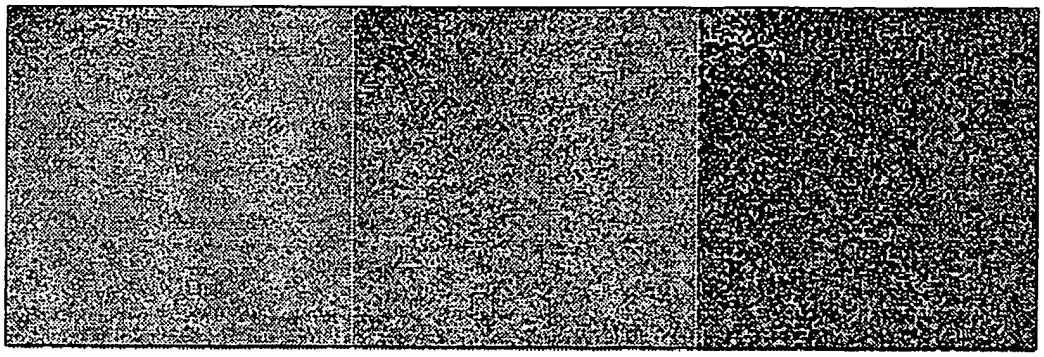


FIGURE 12

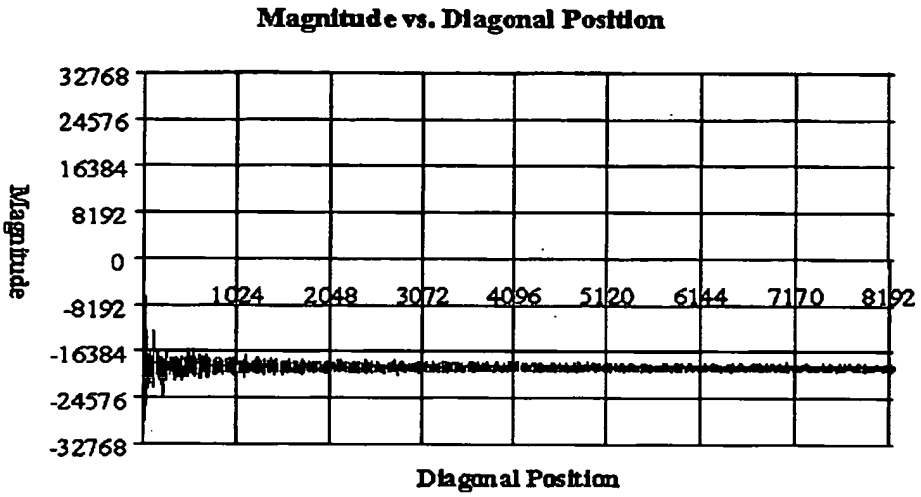


FIGURE 13

Common Mask Dimension	Spectral Filter Size	Included Spectral Coefficients
32 x 32	32	528
64 x 64	38	741
128 x 128	44	990
256 x 256	52	1378
512 x 512	61	1891
1024 x 1024	71	2556

FIGURE 14

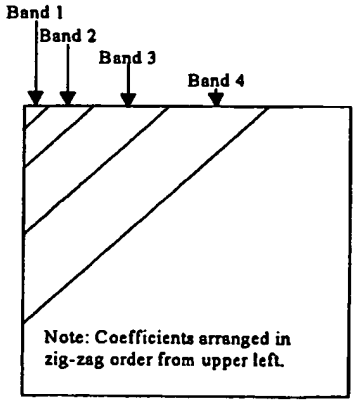


FIGURE 15

Band	Diagonal Rows	Included Spectral Coefficients
1	3	6
2	6	39
3	12	186
4	24	804

FIGURE 16

Mask Spectrum Size	Band 1 Rows	Band 2 Rows	Band 3 Rows	Band 4 Rows
32 x 32	1	2	4	8
64 x 64	2	4	8	16
128 x 128	3	6	12	24
256 x 256	4	8	16	32
512 x 512	5	10	20	40

FIGURE 17

Image Size (pixels)	Image Size (bytes)	Mask Overhead (bytes)	Mask Overhead (%)
64 x 64	4096	546	13.3
128 x 128	16384	764	4.6
256 x 256	65536	1021	1.6
512 x 512	262144	1420	0.5
1024 x 1024	1048576	1948	0.2

FIGURE 18

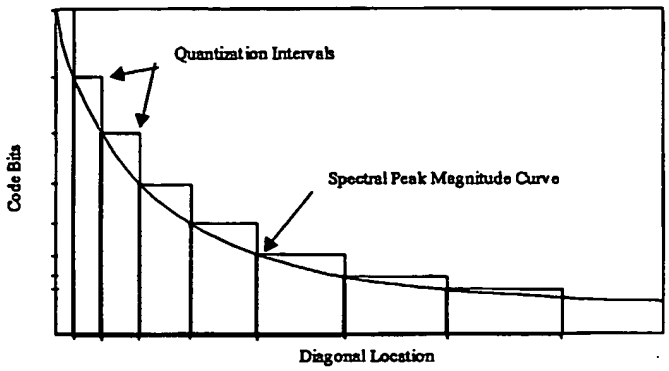


FIGURE 20

FIGURE 21

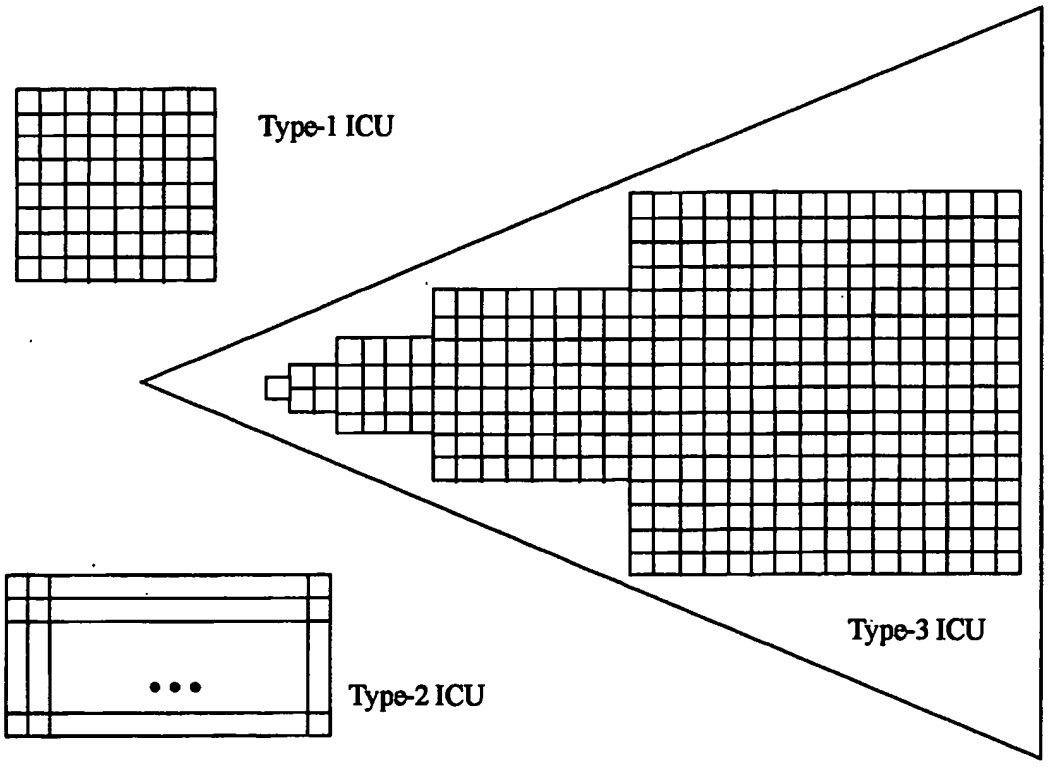


FIGURE 22

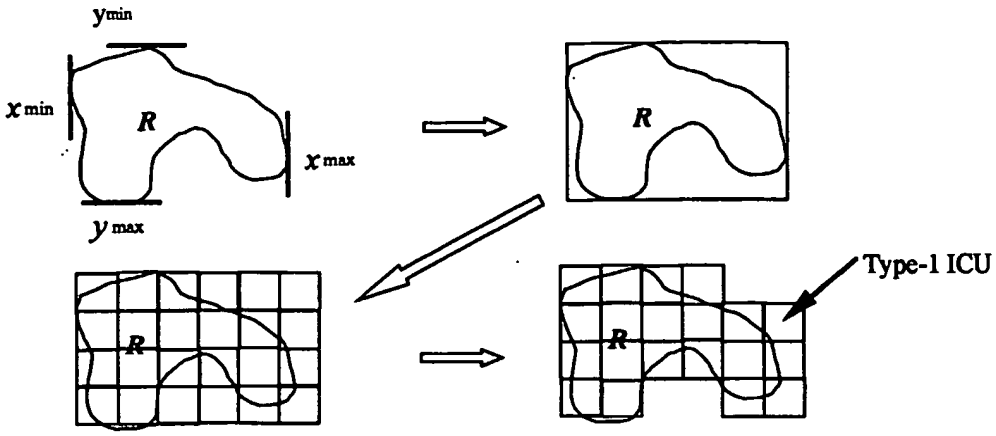


FIGURE 23

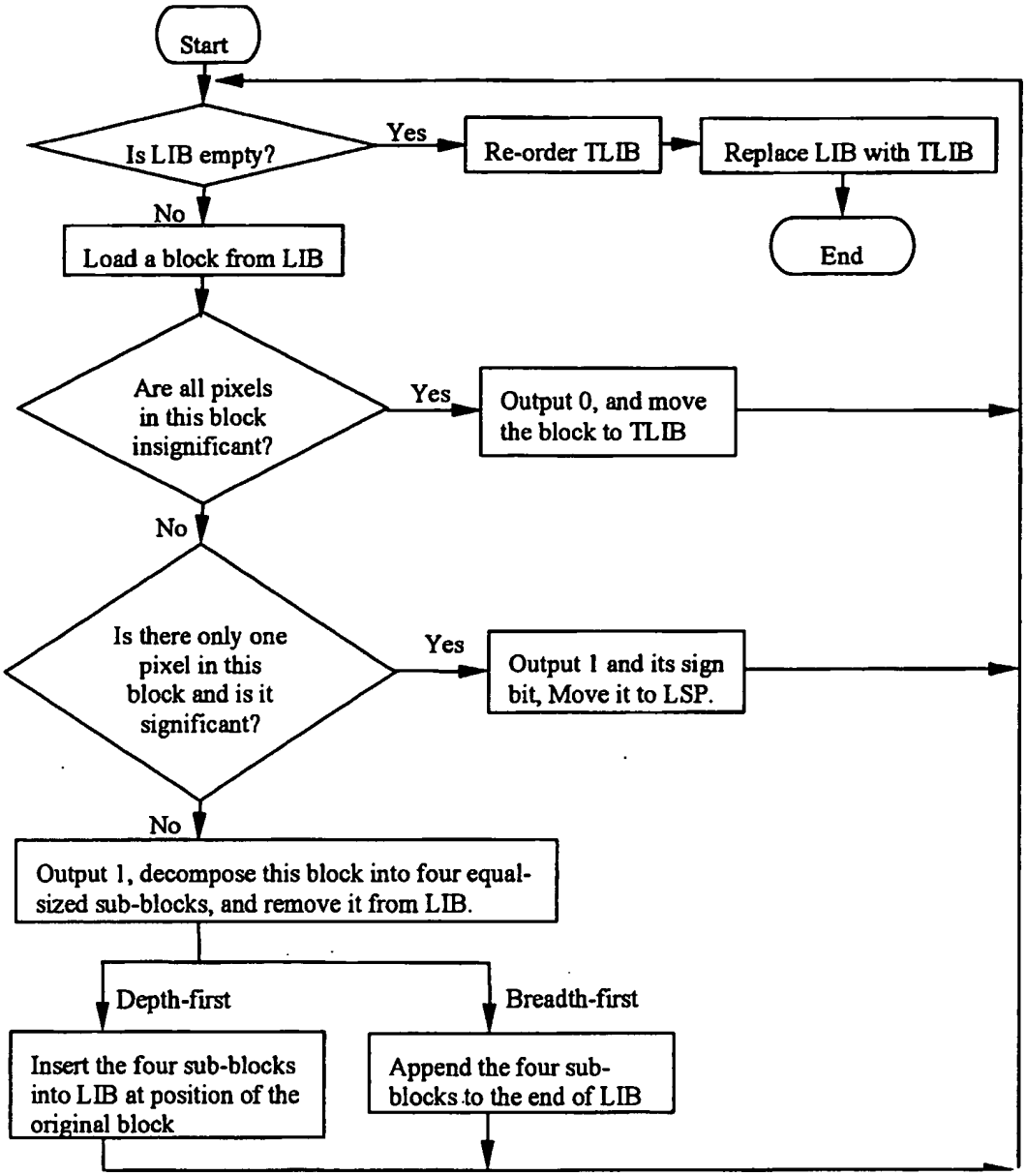

Start
Is LIB empty?
Yes
Re-order TLIB
Replace LIB with TLIB
End
No
Load a block from LIB
Are all pixels in this block insignificant?
Yes
Output 0, and move the block to TLIB
No
Is there only one pixel in this block and is it significant?
Yes
Output 1 and its sign bit, Move it to LSP.
No
Output 1, decompose this block into four equal-sized sub-blocks, and remove it from LIB.
Depth-first
Breadth-first
Insert the four sub-blocks into LIB at position of the original block
Append the four sub-blocks to the end of LIB

FIGURE 24

For LL Subband		For Blocks of Size nxn (n>2) of Other Subbands		For Other Types of Blocks	
Sequence	VLC Code	Sequence	VLC Code	Sequence	VLC Code
1111	000	0001	000	0001	000
1010	001	0010	001	0010	001
0101	010	0100	010	0100	010
0100	011	1111	011	1000	011
1110	1000	0011	1000	1100	1000
1101	1001	1000	1001	1010	1001
0111	1010	1100	1010	0101	1010
0010	1011	1101	1011	0011	1011
1100	11000	0101	11000	0110	11000
1011	11001	0110	11001	0111	11001
1001	11010	0111	11010	1001	11010
1000	11011	1001	11011	1011	11011
0110	11100	1010	11100	1101	11100
0011	11101	1011	11101	1110	11101
0001	11110	1110	11110	1111	11110

FIGURE 25

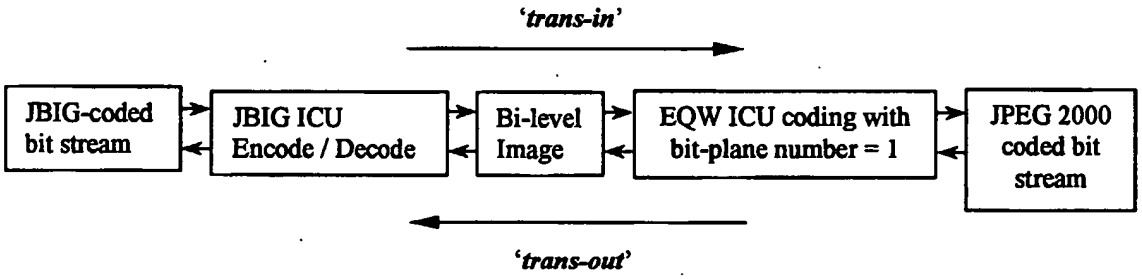


FIGURE 26

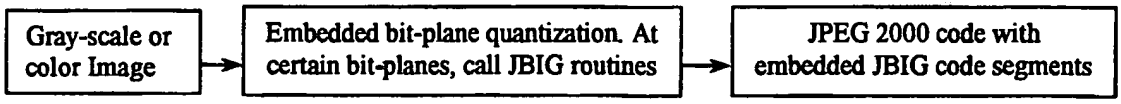
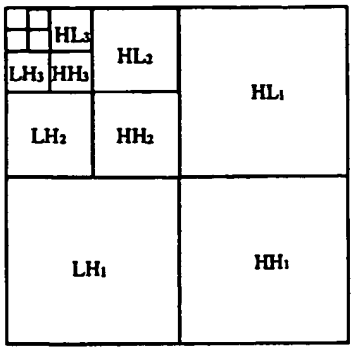
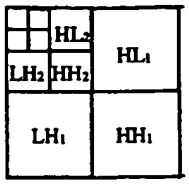


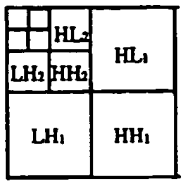
FIGURE 27



Y-channel



U-channel



V-channel

FIGURE 28

Y-channel: LL_4-HL_4-LH_4-HH_4-HL_3-LH_3-HH_3-HL_2-LH_2-HH_2- HL_1-LH_1-HH_1
U-channel: LL_3- HL_3-LH_3-HH_3-HL_2-LH_2-HH_2- HL_1-LH_1-HH_1
V-channel: LL_3- HL_3-LH_3-HH_3-HL_2-LH_2-HH_2-HL_1-LH_1-HH_1

FIGURE 29

Y_{LL4}-Y_{HL4}-Y_{LH4}-Y_{HH4}-U_{LL3}-V_{LL3}-
Y_{HL3}-Y_{LH3}-Y_{HH3}-U_{HL3}-U_{LH3}-U_{HH3}-V_{HL3}-V_{LH3}-V_{HH3}-
Y_{HL2}-Y_{LH2}-Y_{HH2}-U_{HL2}-U_{LH2}-U_{HH2}-V_{HL2}-V_{LH2}-V_{HH2}-
Y_{HL1}-Y_{LH1}-Y_{HH1}-U_{HL1}-U_{LH1}-U_{HH1}-V_{HL1}-V_{LH1}-V_{HH1}

FIGURE 30

Y_{LL4}-Y_{HL4}-Y_{LH4}-Y_{HH4}-U_{LL3}-V_{LL3}-
Y_{HL3}-U_{HL3}-V_{HL3}-Y_{LH3}-U_{LH3}-V_{LH3}-Y_{HH3}-U_{HH3}-V_{HH3}-
Y_{HL2}-U_{HL2}-V_{HL2}-Y_{LH2}-U_{LH2}-V_{LH2}-Y_{HH2}-U_{HH2}-V_{HH2}-
Y_{HL1}-U_{HL1}-V_{HL1}-Y_{LH1}-U_{LH1}-V_{LH1}-Y_{HH1}-U_{HH1}-V_{HH1}

FIGURE 31

Y_{LL4}-U_{LL4}-V_{LL4}-Y_{HL4}-Y_{LH4}-Y_{HH4}-U_{HL4}-U_{LH4}-U_{HH4}-V_{HL4}-V_{LH4}-V_{HH4}-
Y_{HL3}-Y_{LH3}-Y_{HH4}-U_{HL4}-U_{LH4}-U_{HH4}-V_{HL3}-V_{LH3}-V_{HH3}-
Y_{HL2}-Y_{LH2}-Y_{HH2}-U_{HL2}-U_{LH2}-U_{HH2}-V_{HL2}-V_{LH2}-V_{HH2}-
Y_{HL1}-Y_{LH1}-Y_{HH1}-U_{HL1}-U_{LH1}-U_{HH1}-V_{HL1}-V_{LH1}-V_{HH1}

FIGURE 32

Y_{LL4}-U_{LL4}-V_{LL4}-Y_{HL4}- U_{HL4}-V_{HL4}-Y_{LH4}-U_{LH4}-V_{LH4}-Y_{HH4}-U_{HH4}-V_{HH4}
Y_{HL3}-U_{HL3}-V_{HL3}-Y_{LH3}-U_{LH3}-V_{LH3}-Y_{HH3}-U_{HH3}-V_{HH3}
Y_{HL2}-U_{HL2}-V_{HL2}-Y_{LH2}-U_{LH2}-V_{LH2}-Y_{HH2}-U_{HH2}-V_{HH2}
Y_{HL1}-U_{HL1}-V_{HL1}-Y_{LH1}-U_{LH1}-V_{LH1}-Y_{HH1}-U_{HH1}-V_{HH1}

FIGURE 33

Data Type: MUXLIST

Parameters:

liTotBytesPacked	- long integer total bytes packed into the data buffer for this list.
cScheme	- character processing scheme used for data contained in this list.
cHighBit	- character highest bit-level where data processing begins for this list.
*pucMuxBuff	- pointer to unsigned character buffer where data for each bit-level is packed for this list.

Fields for MUX: information for packing after list processing is complete.

pliBitPackInfo[16]	- pointer to long integer number of bits packed into the data buffer at each bit-level for this list.
liCurBytesCount	- long integer current byte count used for bit budget distribution when packing this list.
cCurBitLevel	- character current bit-level used for packing this list.
cRemainingBits	- character remaining bits to be packed at a given bit-level when data to be packed is not evenly divisible by 8 for this list.

FIGURE 34

```
CALCULATE Channel BitBudget // determine optimal bit-budget for each color channel.
INITIALIZE Channel CurrentBitPlane // highest bit plane that exists in each color channel.
INITIALIZE liCurBytesCount and cRemainingBits FOR each MUX list

FOR Each Color Channel // process each channel separately.
    WHILE Channel BitBudget > 0 AND Channel CurrentBitPlane >=0
        FOR Each Wavelet Transform level // beginning at lowest resolution level.
            FOR Each Orientation Set of Data // according to lossy case natural processing order.
                IF cCurBitLevel NOT_EQUAL to Channel CurrentBitPlane
                    CONTINUE
                ELSE
                    SET BitLevelBytes to pliBitPackInfo[Channel CurrentBitPlane] >> 3
                    SET RemBits to pliBitPackInfo[Channel CurrentBitPlane] & 7
                    IF Sum(cRemainingBits, RemBits) >= 8
                        INCREMENT BitLevelBytes by 1
                        DECREMENT cRemainingBits by 8 – RemBits
                    ELSE
                        INCREMENT cRemainingBits by RemBits
                    ENDIF
                    IF Channel BitBudget >= BitLevelBytes
                        INCREMENT liCurBytesCount by BitLevelBytes
                        DECREMENT Channel BitBudget by BitLevelBytes
                        DECREMENT cCurBitLevel by 1
                    ELSE
                        INCREMENT liCurBytesCount by Channel BitBudget
                        SET Channel BitBudget to 0
                    ENDIF
                ENDIF
                IF Channel BitBudget EQUALS 0
                    BREAK
                ENDIF
            END FOR
        END FOR
        DECREMENT Channel CurBitPlane by 1
    END WHILE
END FOR
```

FIGURE 35

Orientation	Max. Data Size (Bytes)	Req. Bits	Req. Bits for High Bit
LL_8	$2x8^2$	7	4
HL_8	$2x8^2$	7	4
LH_8	$2x8^2$	7	4
HH_8	$2x8^2$	7	4
HL_7	$2x16^2$	9	4
LH_7	$2x16^2$	9	4
HH_7	$2x16^2$	9	4
HL_6	$2x32^2$	11	4
LH_6	$2x32^2$	11	4
HH_6	$2x32^2$	11	4
HL_5	$2x64^2$	13	4
LH_5	$2x64^2$	13	4
HH_5	$2x64^2$	13	4
HL_4	$2x128^2$	15	4
LH_4	$2x128^2$	15	4
HH_4	$2x128^2$	15	4
HL_3	$2x256^2$	17	4
LH_3	$2x256^2$	17	4
HH_3	$2x256^2$	17	4
HL_2	$2x512^2$	19	4
LH_2	$2x512^2$	19	4
HH_2	$2x512^2$	19	4
HL_1	$2x1024^2$	21	4
LH_1	$2x1024^2$	21	4
HH_1	$2x1024^2$	21	4
	Total Header Bits	**343**	**100**

FIGURE 36

Image Dimension (Rows & Columns)	Overhead Bits (Lossy)	Overhead Bits (Lossless)	Overhead Bits (GrayScale)
16	44	132	44
32	171	249	83
64	294	384	128
128	435	537	179
256	594	708	236
512	771	897	299
1024	966	1104	368
2048	1179	1329	443
4096	1410	1572	525
8192	1659	1833	611
16384	1926	2112	704
32768	2211	2409	803
65536	2514	2724	908

FIGURE 37

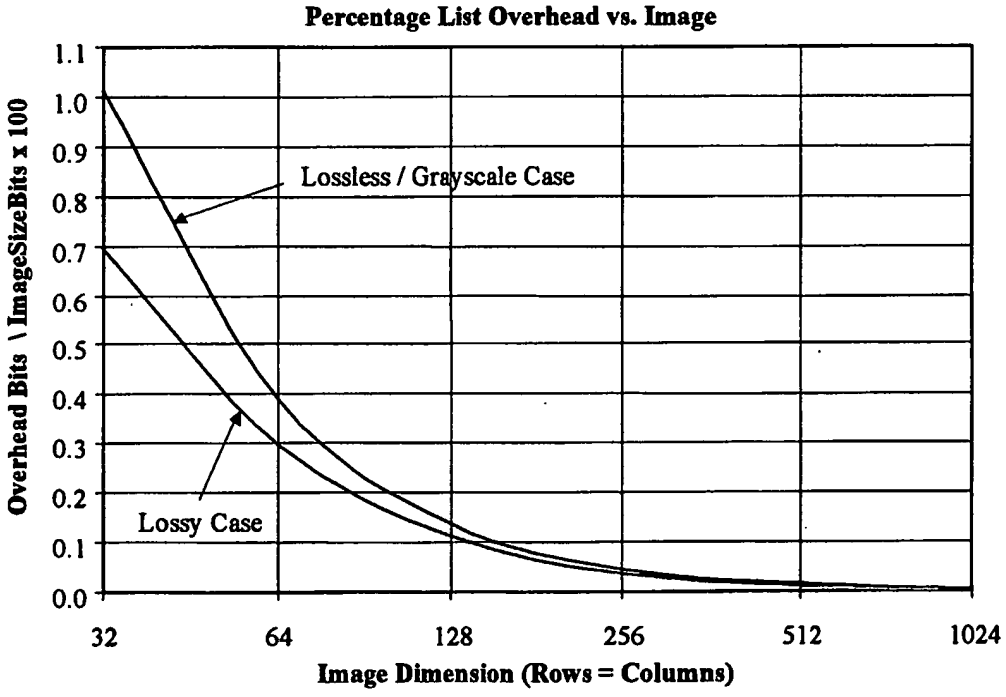


FIGURE 38

Y-channel: $R_1(LL_4\text{-}HL_4\text{-}LH_4\text{-}HH_4\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_2(LL_4\text{-}HL_4\text{-}LH_4\text{-}HH_4\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_3(LL_4\text{-}HL_4\text{-}LH_4\text{-}HH_4\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_4(LL_4\text{-}HL_4\text{-}LH_4\text{-}HH_4\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$.

U-channel: $R_1(LL_3\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_2(LL_3\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_3(LL_3\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_4(LL_3\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$.

V-channel: $R_1(LL_3\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_2(LL_3\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_3(LL_3\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$,
$R_4(LL_3\text{-}HL_3\text{-}LH_3\text{-}HH_3\text{-}HL_2\text{-}LH_2\text{-}HH_2\text{-}HL_1\text{-}LH_1\text{-}HH_1)$.

FIGURE 39

$R_1(Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH3}\text{-}U_{HL3}\text{-}U_{LH3}\text{-}U_{HH3}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1})$,
$R_2(Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH3}\text{-}U_{HL3}\text{-}U_{LH3}\text{-}U_{HH3}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1})$,
$R_3(Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH3}\text{-}U_{HL3}\text{-}U_{LH3}\text{-}U_{HH3}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1})$,
$R_4(Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH3}\text{-}U_{HL3}\text{-}U_{LH3}\text{-}U_{HH3}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1})$.

FIGURE 40

$R_1(Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}U_{HL3}\text{-}V_{HL3}\text{-}Y_{LH3}\text{-}U_{LH3}\text{-}V_{LH3}\text{-}Y_{HH3}\text{-}U_{HH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}U_{HL2}\text{-}V_{HL2}\text{-}Y_{LH2}\text{-}U_{LH2}\text{-}V_{LH2}\text{-}Y_{HH2}\text{-}U_{HH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}U_{HL1}\text{-}V_{HL1}\text{-}Y_{LH1}\text{-}U_{LH1}\text{-}V_{LH1}\text{-}Y_{HH1}\text{-}U_{HH1}\text{-}V_{HH1}),$
$R_2(Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}U_{HL3}\text{-}V_{HL3}\text{-}Y_{LH3}\text{-}U_{LH3}\text{-}V_{LH3}\text{-}Y_{HH3}\text{-}U_{HH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}U_{HL2}\text{-}V_{HL2}\text{-}Y_{LH2}\text{-}U_{LH2}\text{-}V_{LH2}\text{-}Y_{HH2}\text{-}U_{HH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}U_{HL1}\text{-}V_{HL1}\text{-}Y_{LH1}\text{-}U_{LH1}\text{-}V_{LH1}\text{-}Y_{HH1}\text{-}U_{HH1}\text{-}V_{HH1}),$
$R_3(Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}U_{HL3}\text{-}V_{HL3}\text{-}Y_{LH3}\text{-}U_{LH3}\text{-}V_{LH3}\text{-}Y_{HH3}\text{-}U_{HH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}U_{HL2}\text{-}V_{HL2}\text{-}Y_{LH2}\text{-}U_{LH2}\text{-}V_{LH2}\text{-}Y_{HH2}\text{-}U_{HH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}U_{HL1}\text{-}V_{HL1}\text{-}Y_{LH1}\text{-}U_{LH1}\text{-}V_{LH1}\text{-}Y_{HH1}\text{-}U_{HH1}\text{-}V_{HH1}),$
$R_4(Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}U_{HL3}\text{-}V_{HL3}\text{-}Y_{LH3}\text{-}U_{LH3}\text{-}V_{LH3}\text{-}Y_{HH3}\text{-}U_{HH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}U_{HL2}\text{-}V_{HL2}\text{-}Y_{LH2}\text{-}U_{LH2}\text{-}V_{LH2}\text{-}Y_{HH2}\text{-}U_{HH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}U_{HL1}\text{-}V_{HL1}\text{-}Y_{LH1}\text{-}U_{LH1}\text{-}V_{LH1}\text{-}Y_{HH1}\text{-}U_{HH1}\text{-}V_{HH1}).$

FIGURE 41

$R1(Y_{LL4}\text{-}U_{LL4}\text{-}V_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{HL4}\text{-}U_{LH4}\text{-}U_{HH4}\text{-}V_{HL4}\text{-}V_{LH4}\text{-}V_{HH4}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH4}\text{-}U_{HL4}\text{-}U_{LH4}\text{-}U_{HH4}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1}),$
$R2(Y_{LL4}\text{-}U_{LL4}\text{-}V_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{HL4}\text{-}U_{LH4}\text{-}U_{HH4}\text{-}V_{HL4}\text{-}V_{LH4}\text{-}V_{HH4}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH4}\text{-}U_{HL4}\text{-}U_{LH4}\text{-}U_{HH4}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1}),$
$R3(Y_{LL4}\text{-}U_{LL4}\text{-}V_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{HL4}\text{-}U_{LH4}\text{-}U_{HH4}\text{-}V_{HL4}\text{-}V_{LH4}\text{-}V_{HH4}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH4}\text{-}U_{HL4}\text{-}U_{LH4}\text{-}U_{HH4}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1}),$
$R4(Y_{LL4}\text{-}U_{LL4}\text{-}V_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{HL4}\text{-}U_{LH4}\text{-}U_{HH4}\text{-}V_{HL4}\text{-}V_{LH4}\text{-}V_{HH4}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH4}\text{-}U_{HL4}\text{-}U_{LH4}\text{-}U_{HH4}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1}).$

FIGURE 42

$Y_{LL4}\text{-}Y_{HL4}\text{-}Y_{LH4}\text{-}Y_{HH4}\text{-}U_{LL3}\text{-}V_{LL3}\text{-}$
$Y_{HL3}\text{-}Y_{LH3}\text{-}Y_{HH3}\text{-}U_{HL3}\text{-}U_{LH3}\text{-}U_{HH3}\text{-}V_{HL3}\text{-}V_{LH3}\text{-}V_{HH3}\text{-}$
$Y_{HL2}\text{-}Y_{LH2}\text{-}Y_{HH2}\text{-}U_{HL2}\text{-}U_{LH2}\text{-}U_{HH2}\text{-}V_{HL2}\text{-}V_{LH2}\text{-}V_{HH2}\text{-}$
$Y_{HL1}\text{-}Y_{LH1}\text{-}Y_{HH1}\text{-}U_{HL1}\text{-}U_{LH1}\text{-}U_{HH1}\text{-}V_{HL1}\text{-}V_{LH1}\text{-}V_{HH1}$ (R1, R2, R3 , R4).

FIGURE 43

Y_{LL4}-U_{LL4}-V_{LL4}-Y_{HL4}-Y_{LH4}-Y_{HH4}-U_{HL4}-U_{LH4}-U_{HH4}-V_{HL4}-V_{LH4}-V_{HH4}-
Y_{HL3}-Y_{LH3}-Y_{HH4}-U_{HL4}-U_{LH4}-U_{HH4}-V_{HL3}-V_{LH3}-V_{HH3}-
Y_{HL2}-Y_{LH2}-Y_{HH2}-U_{HL2}-U_{LH2}-U_{HH2}-V_{HL2}-V_{LH2}-V_{HH2}-
Y_{HL1}-Y_{LH1}-Y_{HH1}-U_{HL1}-U_{LH1}-U_{HH1}-V_{HL1}-V_{LH1}-V_{HH1} (R1, R2, R3, R4)

FIGURE 44

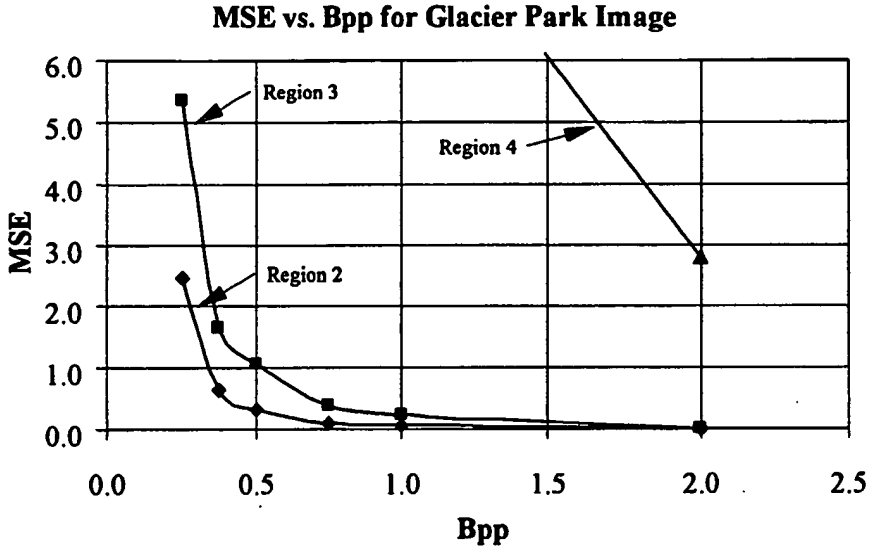


FIGURE 45

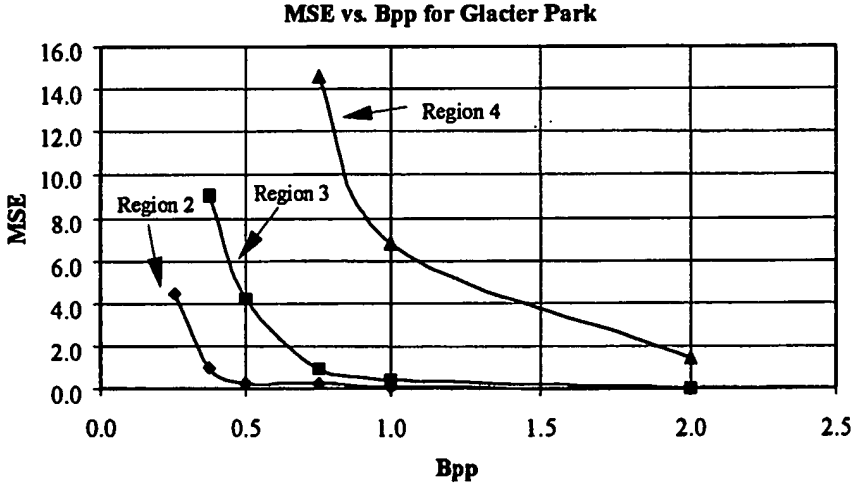


FIGURE 46

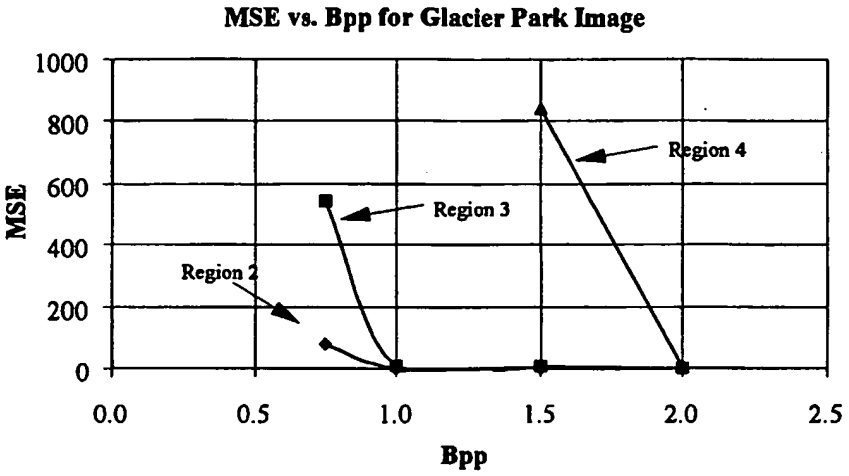


FIGURE 47

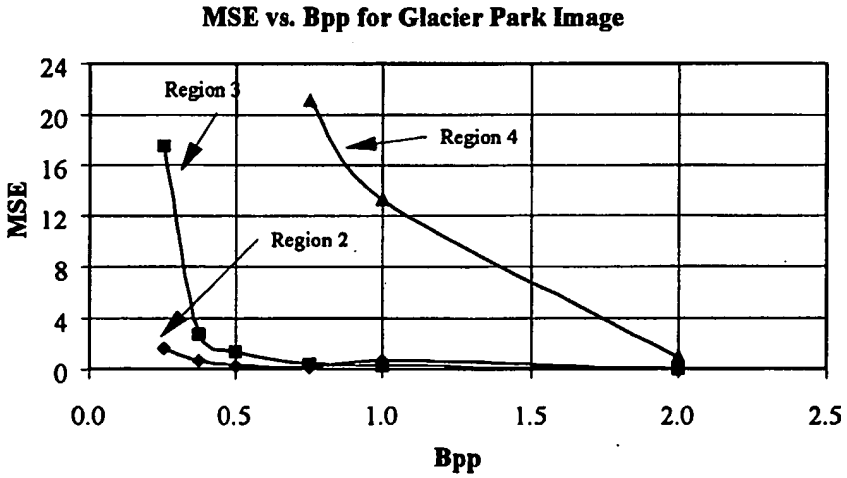


FIGURE 48

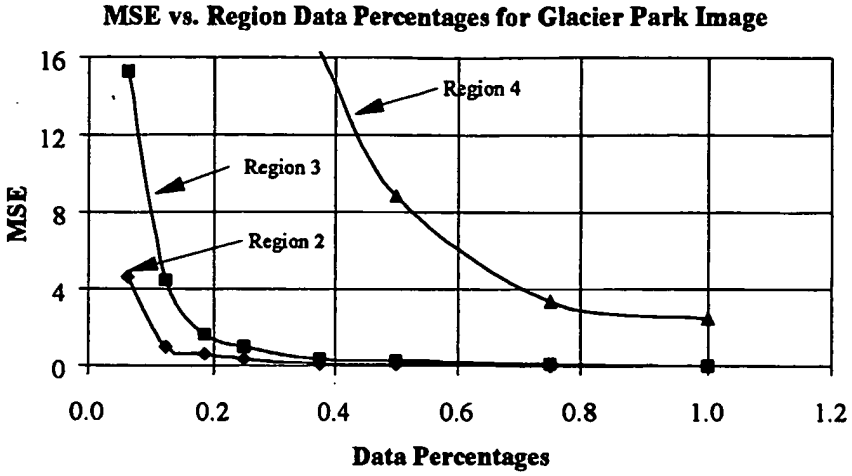


FIGURE 49

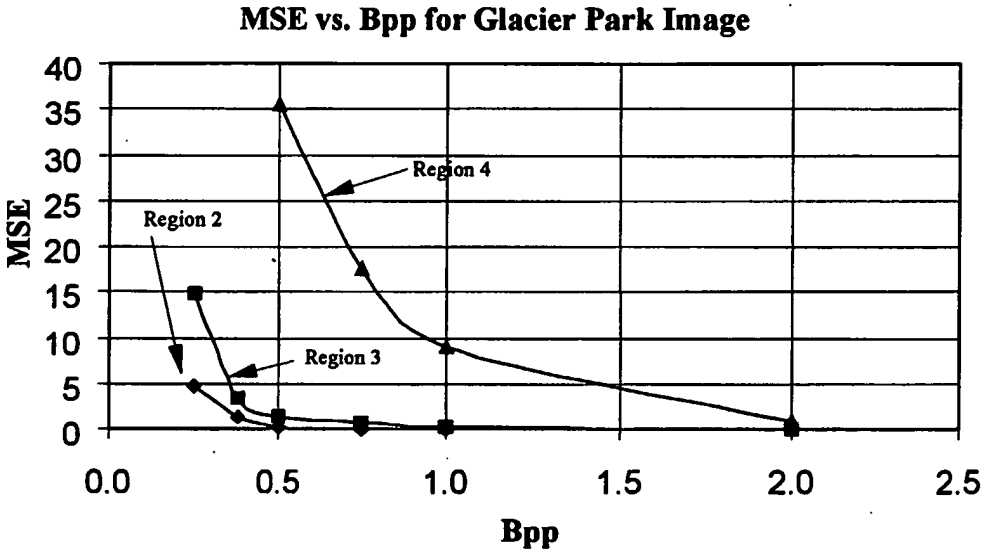


FIGURE 50

Lossless Case		
Packed Item	**bits**	**Selections / Comments**
header size	8	Total Header is normally about 6 bytes
8 bit grayscale / 24 bit color	1	
Image Width	10	(64-1024 columns)
Entropy Coding	2	None, Basic, Adaptive
Image Height	10	(64-1024 rows)
Wavelet Transform Levels	3	Level >= 1
Region Channels	2	1, 2, 3, 4
Mask Type	2	None, User, Auto (DCT)
Pack Raw Mask Flag	1	Read in file for mask
Mask Procedure	3	DCT Common Mask, Raw Common Mask with VM Translation,
Down Sampling Type	2	Heuristic DCT Down Sampling
Region Start Level	4	No Regions to # wavelet levels
Lossy Flag	1	Lossless
Color Transform Type	2	2 Internal, YIQ (full data sets)
Sort Type	1	1D, EQW
Wavelet Kernel	2	Lifting Scheme

FIGURE 51

Lossy Case

Packed Item	bits	Selections
header size	8	Between 6 to 12 bytes
8 bit grayscale / 24 bit color	1	
Image Width	10	(64-1024)
Image Height	10	(64-1024)
Wavelet Transform Levels	3	L >= 2
Region Channels	2	1, 2, 3, 4
Mask Type	2	None, User, Auto (DCT)
Pack Raw Mask Flag	1	Read in file for mask
Mask Procedure	3	DCT Common Mask, Raw Common Mask with VM Resolution Translation, User primitives Common or Full Size, User arbitrarily defined in Common and Full sizes
Down Sampling Type	2	Heuristic DCT Down Sampling
Region Start Level	4	No Regions to #wavelet levels
Lossy Flag	1	lossy
Color Transform Type	2	2 Internal, KL, YIQ, YUV (down sampled data sets)
Post Filter	1	Under Implementation
Sort Type	1	1D, EQW
Wavelet Kernel	3	Daubaches, Symlet, Coiflet, Biorthogonal, Lifting Schemes
Filter Type	4	Various common filter types/sizes
KL Color Transform	48	(if enabled)

FIGURE 52

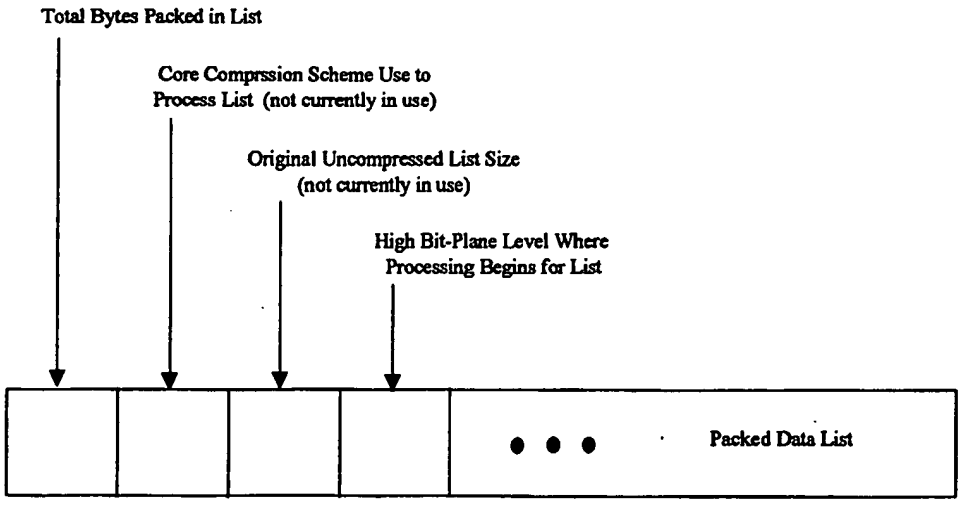


FIGURE 53

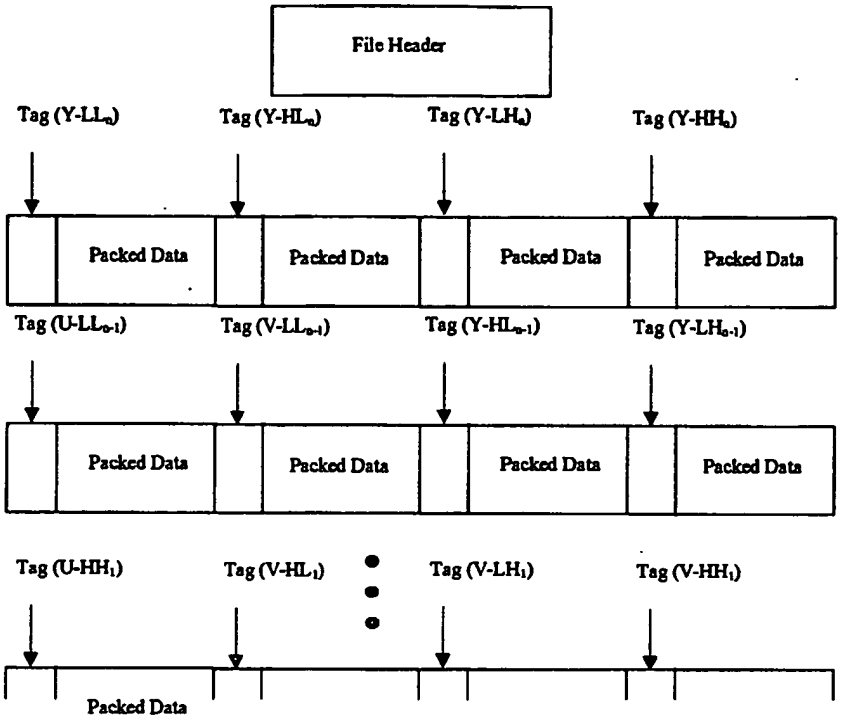

File Header
Tag (Y-LL$_n$)
Tag (Y-HL$_n$)
Tag (Y-LH$_n$)
Tag (Y-HH$_n$)
Packed Data
Packed Data
Packed Data
Packed Data
Tag (U-LL$_{n-1}$)
Tag (V-LL$_{n-1}$)
Tag (Y-HL$_{n-1}$)
Tag (Y-LH$_{n-1}$)
Packed Data
Packed Data
Packed Data
Packed Data
Tag (U-HH$_1$)
Tag (V-HL$_1$)
Tag (V-LH$_1$)
Tag (V-HH$_1$)
Packed Data

FIGURE 54

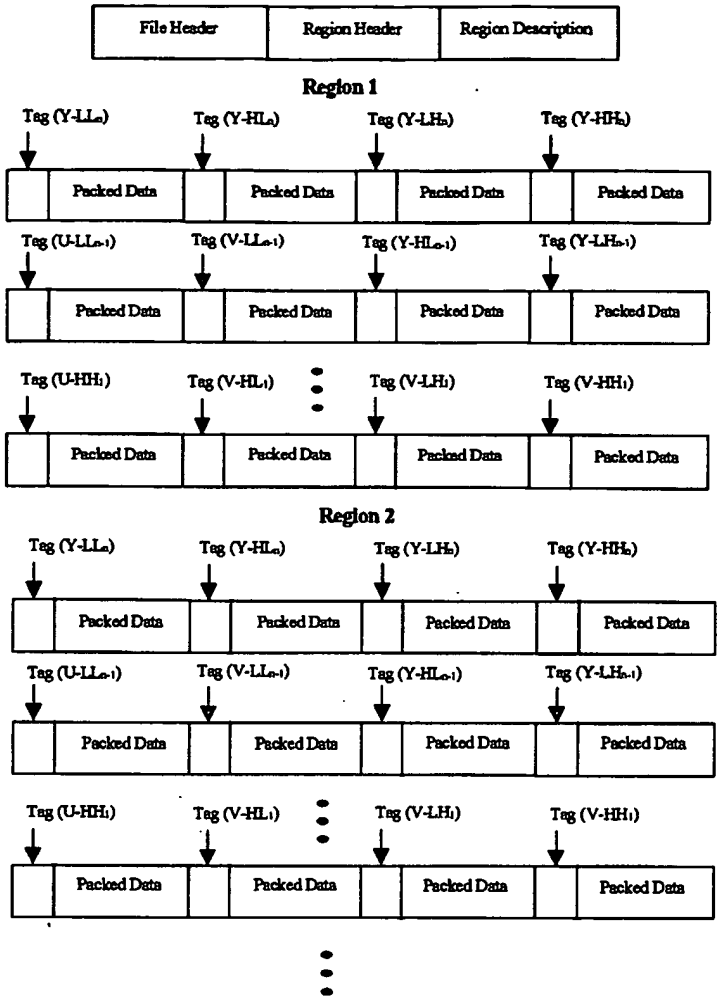

File Header
Region Header
Region Description
Region 1
Tag (Y-LLn)
Tag (Y-HLn)
Tag (Y-LHn)
Tag (Y-HHn)
Packed Data
Tag (U-LLn-1)
Tag (V-LLn-1)
Tag (Y-HLn-1)
Tag (Y-LHn-1)
Tag (U-HH1)
Tag (V-HL1)
Tag (V-LH1)
Tag (V-HH1)
Region 2
Tag (Y-LLn)
Tag (Y-HLn)
Tag (Y-LHn)
Tag (Y-HHn)
Tag (U-LLn-1)
Tag (V-LLn-1)
Tag (Y-HLn-1)
Tag (Y-LHn-1)
Tag (U-HH1)
Tag (V-HL1)
Tag (V-LH1)
Tag (V-HH1)

FIGURE 55

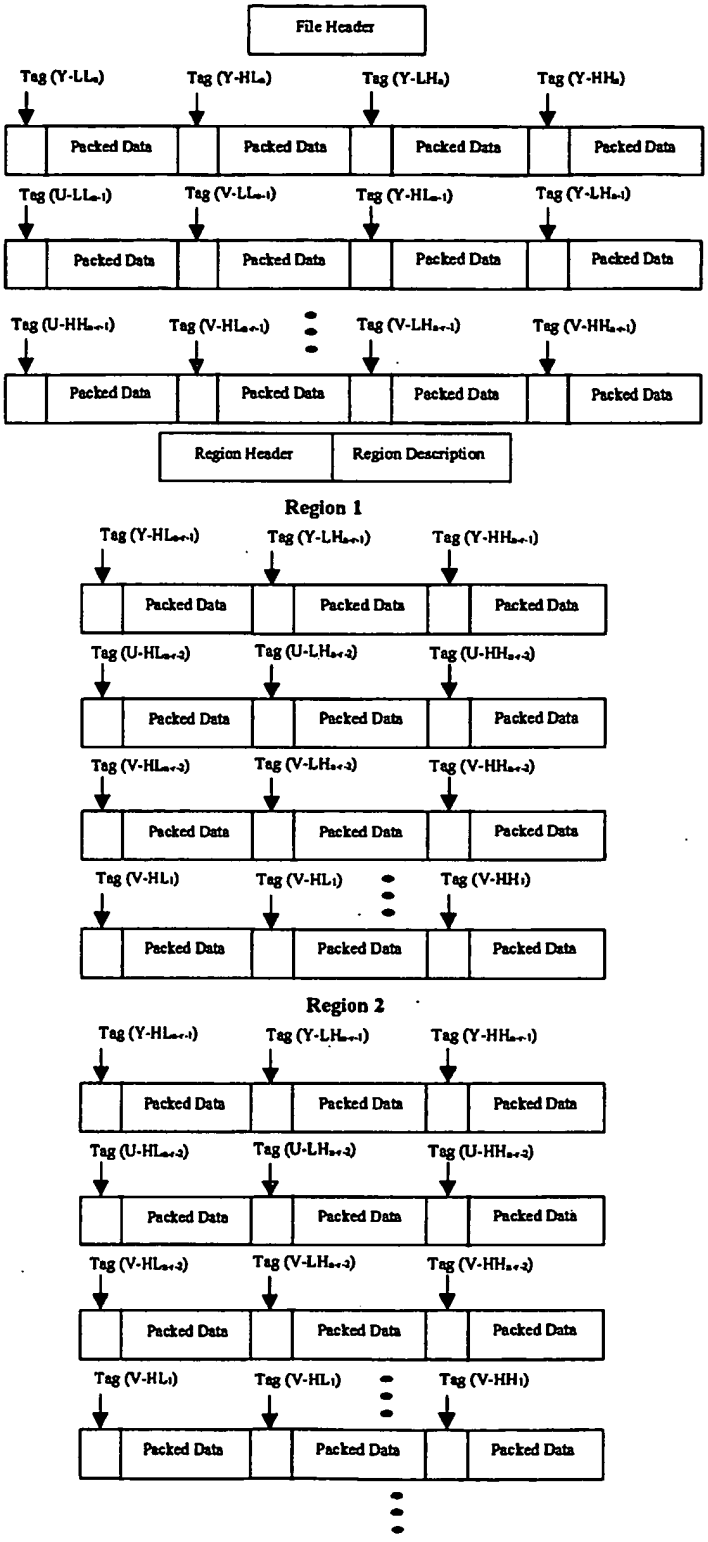

File Header
Tag (Y-LLn)
Tag (Y-HLn)
Tag (Y-LHn)
Tag (Y-HHn)
Packed Data
Tag (U-LLn-1)
Tag (V-LLn-1)
Tag (Y-HLn-1)
Tag (Y-LHn-1)
Tag (U-HHn-1)
Tag (V-HLn-1)
Tag (V-LHn-1)
Tag (V-HHn-1)
Region Header
Region Description
Region 1
Tag (Y-HLn-1)
Tag (Y-LHn-1)
Tag (Y-HHn-1)
Tag (U-HLn-2)
Tag (U-LHn-2)
Tag (U-HHn-2)
Tag (V-HLn-2)
Tag (V-LHn-2)
Tag (V-HHn-2)
Tag (V-HL1)
Tag (V-HL1)
Tag (V-HH1)
Region 2

FIGURE 56

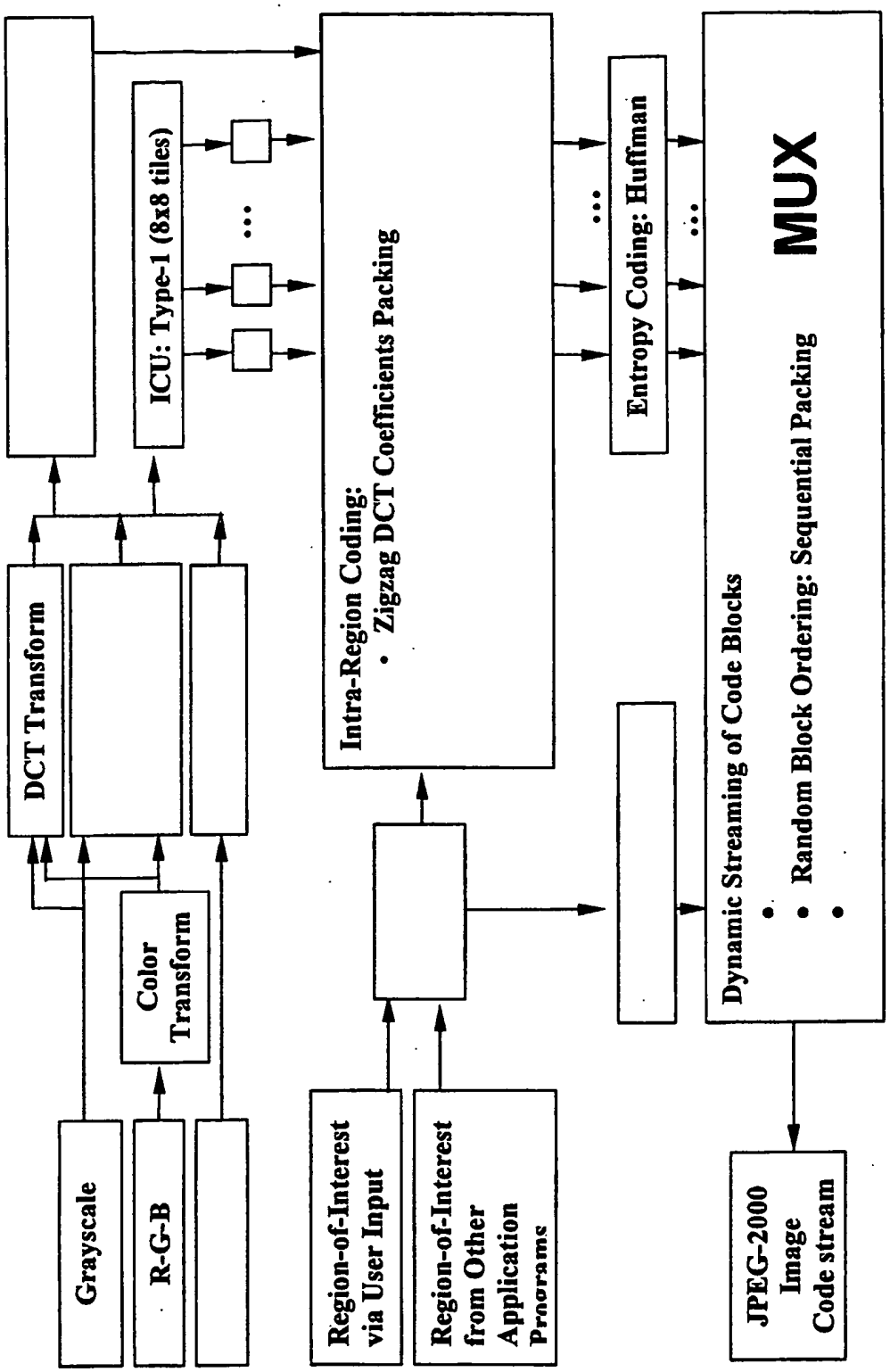

Grayscale
R-G-B
DCT Transform
Color Transform
ICU: Type-1 (8x8 tiles)
Region-of-Interest via User Input
Region-of-Interest from Other Application Programs
Intra-Region Coding:
• Zigzag DCT Coefficients Packing
Entropy Coding: Huffman
Dynamic Streaming of Code Blocks
• Random Block Ordering; Sequential Packing
MUX
JPEG-2000 Image Code stream

FIGURE 57

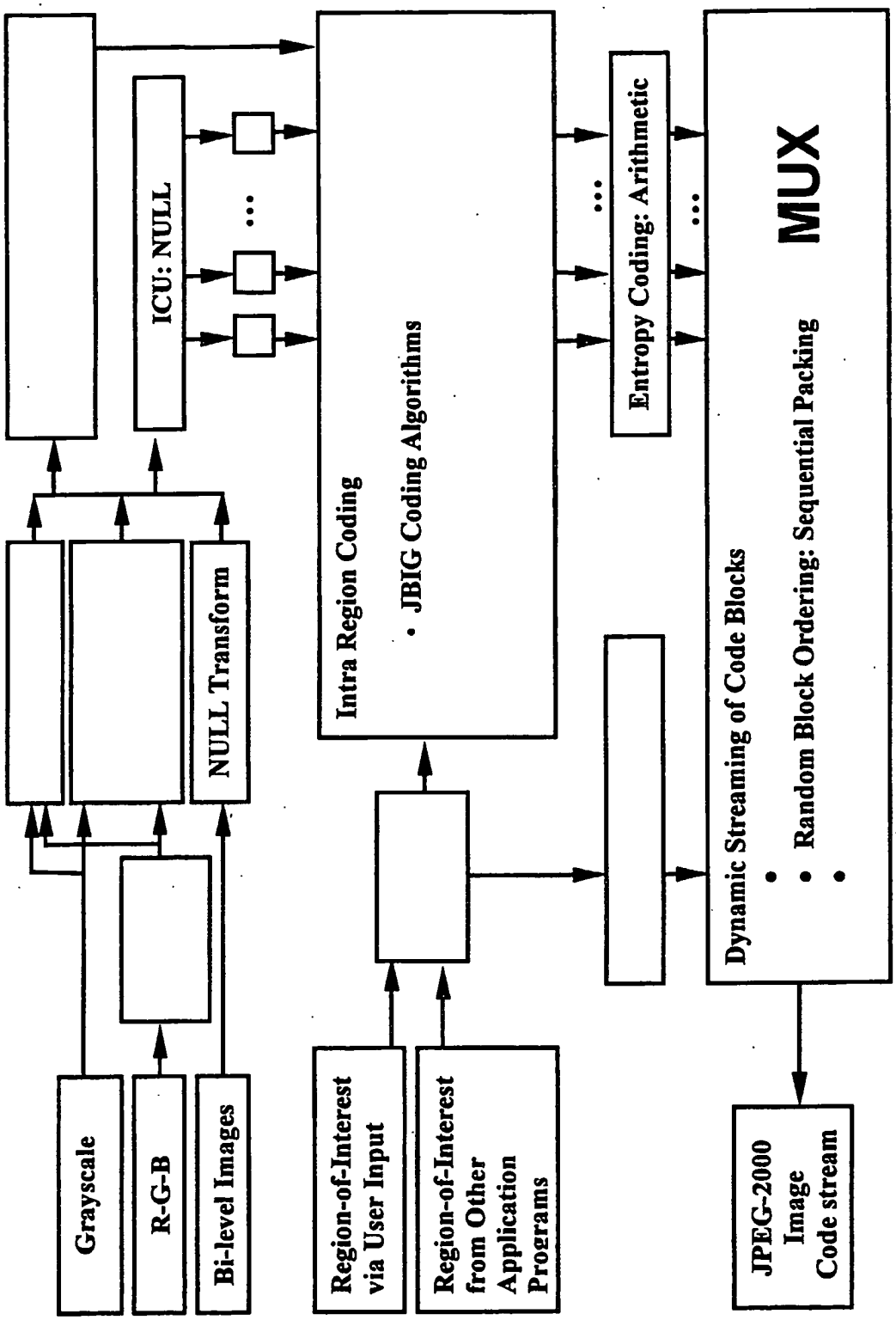

Grayscale
R-G-B
Bi-level Images
NULL Transform
ICU: NULL
Intra Region Coding
• JBIG Coding Algorithms
Entropy Coding: Arithmetic
Dynamic Streaming of Code Blocks
• Random Block Ordering: Sequential Packing
MUX
Region-of-Interest via User Input
Region-of-Interest from Other Application Programs
JPEG-2000 Image Code stream

FIGURE 58

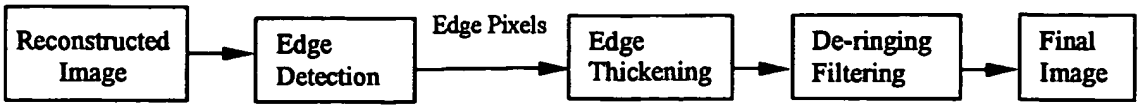
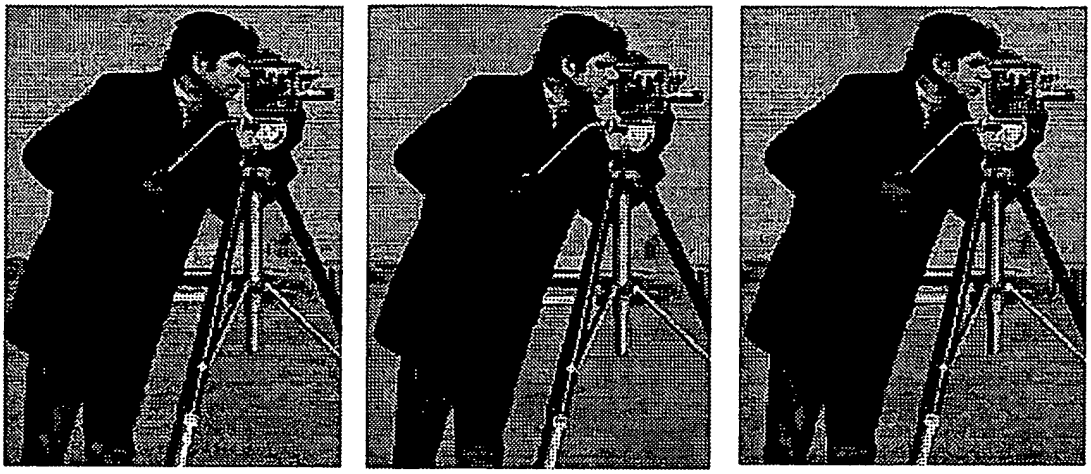


FIGURE 59



Re-constructed Image VM 3.0 (B) Post-Processing Adaptive Post-Processing

FIGURE 60

FIGURE 61

Compression ratio	VM 3.0 (B) time (sec)	RICS Adaptive time (sec)	Compression ratio	VM 3.0 (B) time (sec)	RICS Adaptive time (sec)
3.2170	7.310	1.743	3.709	22.392	5.398
9.0140	7.260	1.702	8.889	23.023	8.151
40.210	7.325	1.482	10.000	22.442	5.288
48.763	7.170	1.462	11.428	22.232	5.809
56.492	8.262	1.402	13.333	22.462	5.878
66.550	8.703	1.382	16.000	22.482	5.629
81.143	6.769	1.362	20.001	22.522	5.489
103.716	7.171	1.175	26.666	22.382	5.568
130.168	7.170	1.222	32.000	22.422	5.338
169.019	8.310	1.132	40.002	22.352	3.896
253.337	7.130	1.052	79.987	22.282	3.465

FIGURE 62

Compression Ratio	PSNR (γ=8)	PSNR (γ=12)	PSNR (γ=16)
3.709	31.329611	31.177579	30.511348
4.000	31.300452	31.154471	30.490295
5.333	31.157031	30.994670	30.357978
8.000	30.630383	30.474574	29.862022
8.889	30.401649	30.216825	29.646734
10.000	30.067778	29.909074	29.375402
11.428	29.691347	29.546710	29.056083
13.333	29.228454	29.103298	28.645196
16.000	28.407724	28.276664	27.908937
20.001	27.508670	27.413509	27.134810
26.666	26.385419	26.307257	26.084268
32.000	25.333808	25.280108	25.111808
40.002	24.696655	24.660851	24.509536
79.987	22.179274	22.152787	22.093103
100.004	21.592595	21.568550	21.525140

FIGURE 63

Compression Ratio	PSNR (F=3)	PSNR (F=5)	PSNR (F=7)	PSNR (F=9)	PSNR (F=11)
1.583	35.514592	34.206867	33.687448	33.409849	33.223806
1.600	35.516204	34.201801	33.674565	33.408178	33.220934
2.000	35.314595	34.080237	33.538998	33.301232	33.090358
2.667	34.957612	33.765719	33.291270	33.011713	32.832969
4.000	34.002797	32.997735	32.564248	32.338257	32.200522
8.000	31.089735	30.554027	30.324381	30.196463	30.106473
8.889	30.572145	30.142348	29.935209	29.796335	29.725453
9.999	30.131952	29.795631	29.604549	29.472401	29.394763
11.429	29.629346	29.390456	29.261471	29.13131	29.064764
13.334	28.854873	28.701271	28.602145	28.527964	28.470308
16.000	27.870101	27.760478	27.671142	27.632609	27.579655
19.999	27.031806	26.910914	26.857123	26.835492	26.794267
26.662	25.962692	25.994100	25.999759	26.019787	25.991909
40.010	24.159134	24.178761	24.173011	24.191618	24.182049
80.020	21.654340	21.683920	21.731559	21.761676	21.774576

FIGURE 64

Compression Ratio	PSNR (F=3)	PSNR (F=5)	PSNR (F=7)	PSNR (F=9)	PSNR (F=11)
3.709	31.656960	30.740420	30.806267	30.511348	30.298514
4.000	31.634763	30.735842	30.791791	30.490295	30.278695
5.333	31.436411	30.558898	30.635121	30.357978	30.141580
8.000	30.819587	30.065789	30.126713	29.862022	29.680912
8.889	30.537579	29.839074	29.879854	29.646734	29.474362
10.000	30.170406	29.541099	29.603527	29.375402	29.205815
11.428	29.740858	29.186726	29.255123	29.056083	28.880456
13.333	29.255225	28.752598	28.839294	28.645196	28.494299
16.000	28.391715	27.982943	28.075866	27.908937	27.788081
20.001	27.479117	27.162369	27.270921	27.134810	27.012976
26.666	26.329007	26.079132	26.175590	26.084268	25.999266
32.000	25.279960	25.088255	25.181387	25.111808	25.044080
40.002	24.649334	24.486896	24.568579	24.509536	24.446939
79.987	22.195351	22.099853	22.127757	22.093103	22.066329
100.004	21.598939	21.520845	21.561808	21.525140	21.501686

FIGURE 65

Comp Ratio	MSE C4	PSNR C4	MSE C6	PSNR C6	MSE C8	PSNR C8	MSE C10	PSNR C10
1.583	7.765411	39.229159	14.264114	36.588355	20.262634	35.063845	24.678070	34.207692
1.600	7.783203	39.219220	14.271622	36.586070	20.312759	35.053114	24.757339	34.193764
2.000	8.327881	38.925459	14.998596	36.370297	21.123337	34.883178	25.556641	34.055766
2.667	9.692001	38.266669	16.493225	35.957748	22.997864	34.513929	27.498306	33.737744
4.000	14.722046	36.451122	21.984375	34.709662	28.808853	33.535544	33.097488	32.932853
8.000	44.483109	31.648852	49.900986	31.149712	55.549133	30.684031	59.073853	30.416851
8.889	52.799194	30.904531	57.667984	30.521456	62.367477	30.181222	65.356079	29.977944
9.999	60.632019	30.303783	64.887344	30.009204	68.774963	29.756500	71.202606	29.605845
11.429	69.232758	29.727687	72.769318	29.511321	75.513184	29.350576	77.597488	29.232327
13.334	82.567825	28.962695	85.462585	28.813043	87.943863	28.688748	89.701625	28.602798
16.000	105.474950	27.899311	108.281906	27.785245	109.973099	27.717939	111.452667	27.659899
19.999	131.348300	26.946559	133.166946	26.886839	134.350037	26.848426	134.844223	26.832480
26.662	166.120390	25.926574	165.26181	25.949079	164.059326	25.980794	163.365967	25.999188
40.010	249.851910	24.153977	249.164856	24.165936	248.268158	24.181593	247.954636	24.187081
80.020	439.120450	21.704967	437.621460	21.719818	435.972107	21.736217	434.750366	21.748404

FIGURE 66

Comp Ratio	MSE C12	PSNR C12	MSE C14	PSNR C14	MSE C16	PSNR C16
1.583	29.654851	33.409849	35.271851	32.656521	41.555957	31.965423
1.600	29.666245	33.408178	35.268005	32.656995	41.379425	31.962959
2.000	30.405853	33.301232	36.249649	32.537766	42.367615	31.860463
2.667	32.501923	33.011713	38.265869	32.302688	43.982712	31.697984
4.000	37.953751	32.338257	43.370041	31.758905	48.739197	31.252020
8.000	62.148987	30.196463	65.897278	29.942129	69.077271	29.737452
8.889	68.147018	29.796335	70.858078	29.626910	73.575455	29.463474
9.999	73.424377	29.472401	75.886185	29.329176	78.898651	29.160108
11.429	79.423569	29.131310	81.619110	29.012885	84.361023	28.869385
13.334	91.260742	28.527964	93.195953	28.436833	95.293335	28.340178
16.000	112.155212	27.632609	113.068161	27.597400	114.424561	27.545611
19.999	134.750748	26.835492	135.182816	26.821589	135.73201	26.803981
26.662	162.592941	26.019787	162.833038	26.013378	163.141922	26.005148
40.010	247.695740	24.191618	248.170792	24.183297	249.198486	24.165350
80.020	433.423813	21.761676	433.733047	21.758578	434.813171	21.747777

FIGURE 67

Comp Ratio	MSE C6	PSNR C6	MSE C8	PSNR C8	MSE C10	PSNR C10
3.709	40.866048	32.017177	48.421931	31.280383	53.364817	30.858253
4.000	41.139272	31.988238	48.628245	31.261918	53.618078	30.837691
5.333	43.182419	31.777734	50.741887	31.077137	55.548584	30.684074
8.000	50.567963	31.092049	57.770162	30.513768	62.558919	30.167911
8.889	54.322367	30.781017	61.445058	30.245934	65.941528	29.939214
10.000	59.253784	30.403643	66.186651	29.923100	70.753092	29.633349
11.428	65.823085	29.947021	72.189621	29.546056	76.476232	29.295539
13.333	73.587957	29.462736	80.245789	29.086581	84.515152	28.861458
16.000	90.881770	28.546036	97.184072	28.254853	101.208043	28.078653
20.001	113.080653	27.596921	118.35673	27.398874	122.169271	27.261184
26.666	147.815740	26.433597	152.951772	26.285258	156.646398	26.181599
32.000	189.147156	25.362805	193.993169	25.252939	197.467316	25.175851
40.002	219.816732	24.710196	224.114268	24.626109	227.036153	24.569853
79.987	388.352183	22.238546	393.958450	22.176299	397.818293	22.133956
100.004	446.311579	21.634422	450.888418	21.590113	454.329178	21.557097

FIGURE 68

Comp Ratio	PSNR R1	PSNR R2	PSNR R3	PSNR R4	PSNR R5
1.583	36.574570	34.642943	33.409849	32.697519	32.241311
1.600	36.562052	34.637526	33.408178	32.690100	32.238019
2.000	36.362537	34.486701	33.301232	32.599597	32.146102
2.667	35.932465	34.134055	33.011713	32.347816	31.926978
4.000	34.722586	33.301598	32.338257	31.778517	31.416868
8.000	31.264379	30.640440	30.196463	29.892131	29.673122
8.889	30.660431	30.183697	29.796335	29.547117	29.365451
9.999	30.201141	29.800439	29.472401	29.244209	29.090109
11.429	29.706536	29.398255	29.131310	28.942109	28.805907
13.334	28.946950	28.733637	28.527964	28.378407	28.263955
16.000	27.924645	27.784119	27.632609	27.510373	27.424183
19.999	27.018210	26.938891	26.835492	26.749663	26.687918
26.662	26.025142	26.045564	26.019787	25.971153	25.921551
40.010	24.209514	24.214338	24.191618	24.152663	24.116273
80.020	21.70436	21.747276	21.761676	21.762267	21.751451

FIGURE 69

Parameter	Number of Bits	Value Range
Mask Threshold	7	0-127
Mask Width	4	5-20
Estimation Threshold	4	5-20
Filter Length	3	3-10
Constraints	4	3-18
Iteration	2	1-4

FIGURE 70